Final Course Project Presentation

Mengye Ren

NYU

Dec 12, 2023

Final Course Evaluation

- Please share your feedback with us and help us improve.

Logistics

- 22 groups submitted their slides
- Topics vary across many fields: finance, imaging, biology, e-commerce, security, etc.
- Aim your talk around 3 minutes. Hard stop at 4 minutes
- How to show good respect to presenters? Ask good questions! (participation score)
- Save your question at the end of each presentation!

Final Report

- Submit your code and report PDF as a zip file.
- Due: Dec 15 11:59pm
- Use the LaTeX template from the course website!
  https://nyu-cs2565.github.io/2023-fall/#project
- Instructions and rubrics on the course website.

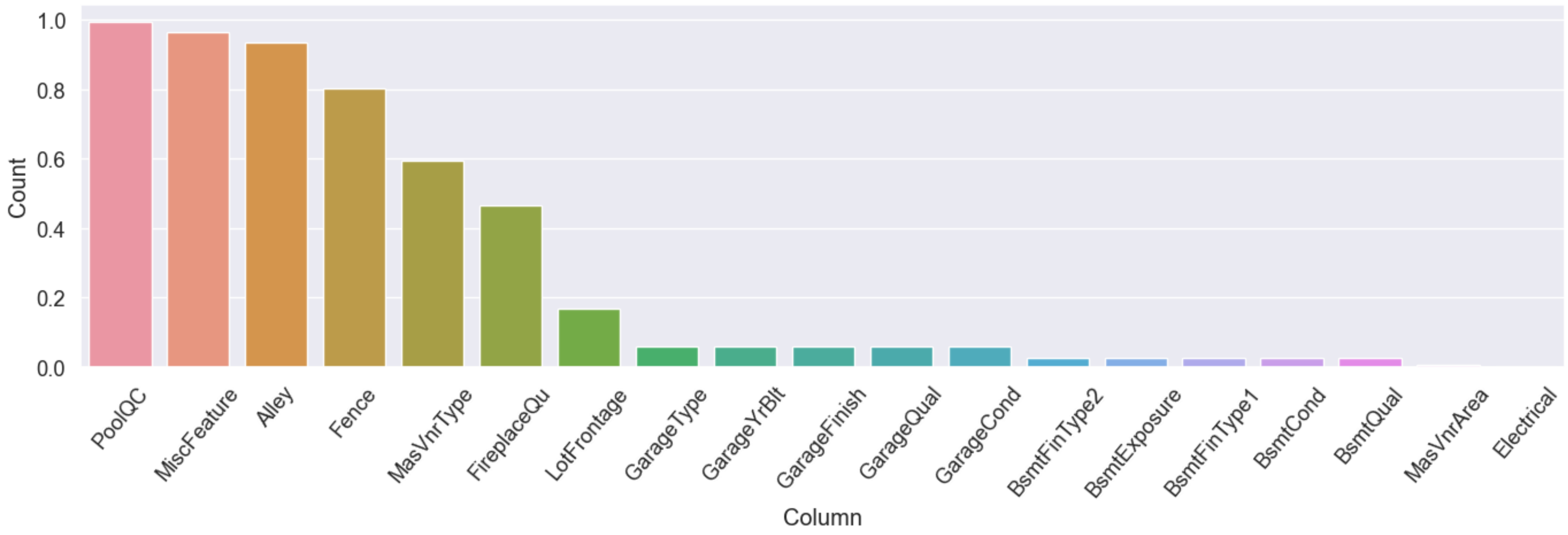
Real Estate Price Prediction with ML Techniques

Dataset

- https://www.kaggle.com/datasets/lespin/house-prices-dataset/data
- Each row contains 79 features to describe the condition of the house, including numeric features, such as numbers of bathrooms, bedrooms, living rooms, lot size, etc. and categorical features including zoning classification, all kinds of condition info, etc.
- 1460 datapoints in total, including missing data and wrong data.

Missing Data

- `missing_data = train.isna().sum() / train.shape[0]`
- `sns.barplot(data=missing_data, x='Column', y='Count')`



Dealing with Missing Data

- For the columns with more than 50% of missing data, drop them.
- For numeric data: For LotFrontage, which is linear feet of street connected to property, it is a high probability that these values are similar to houses in the same Neighborhood, so fill them with the median value in the same Neighborhood. For MasVnrArea, fill with 0 and for GarageYrBlt, fill with the median value of the dataset.

Dealing with Missing Data

- For the rest 12 columns of catagorical data: fill them with NA, No or other typical values. Map dic is shown as below.

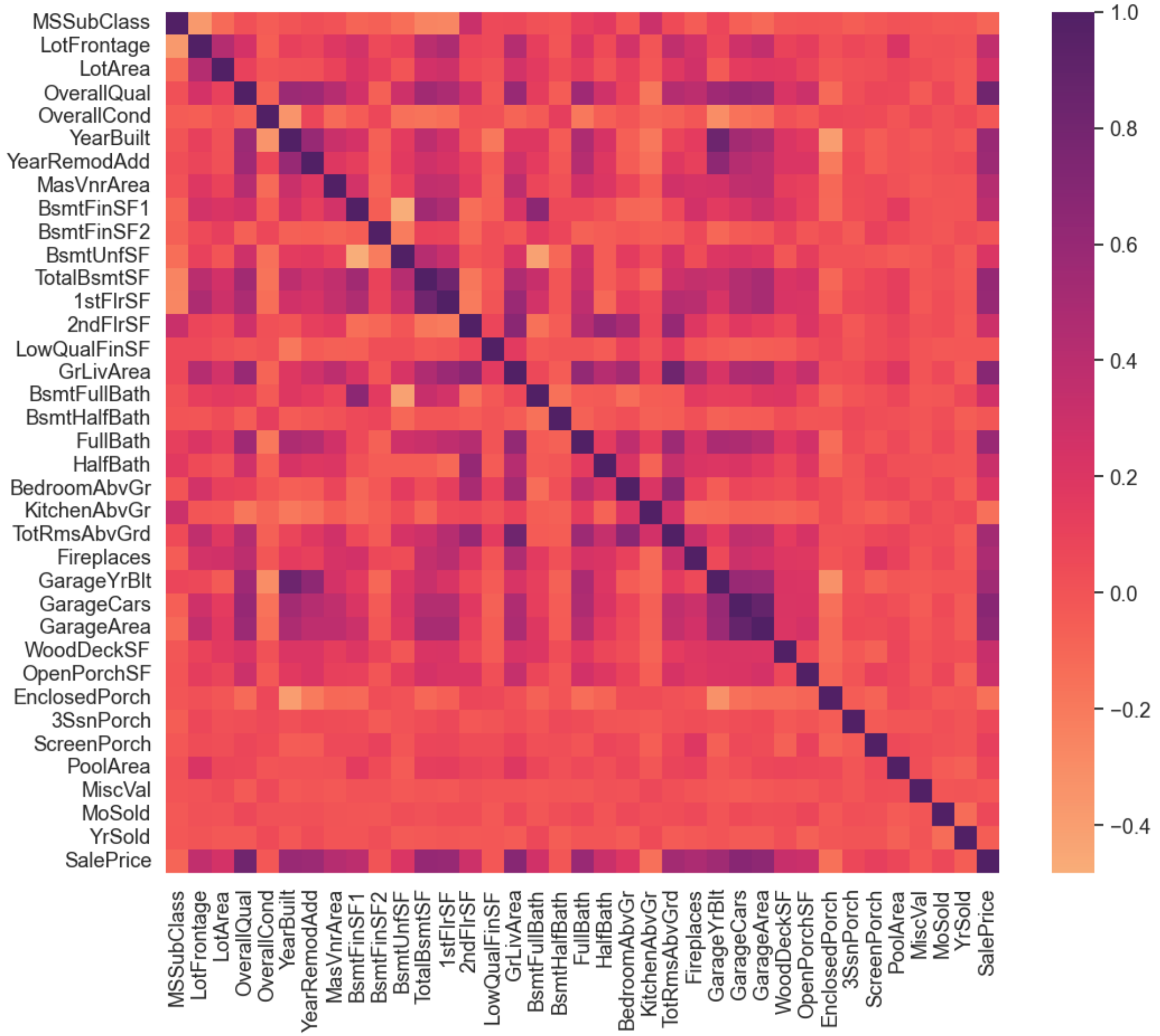
```
none_conversion = [
        ("MasVnrType", "None"), ("Electrical", "SBrkr"), ("BsmtQual", "NA"),
        ("BsmtCond", "TA"), ("BsmtExposure", "No"), ("BsmtFinType1", "No"),
        ("BsmtFinType2", "No"), ("FireplaceQu", "NA"), ("GarageType", "No"),
        ("GarageFinish", "No"), ("GarageQual", "NA"), ("GarageCond", "NA"),
]
```

Feature Engineering

- For numeric features: Draw the corr heatmap and analyse each feature.

Feature Engineering

- For numeric fea… …lyse each feature.

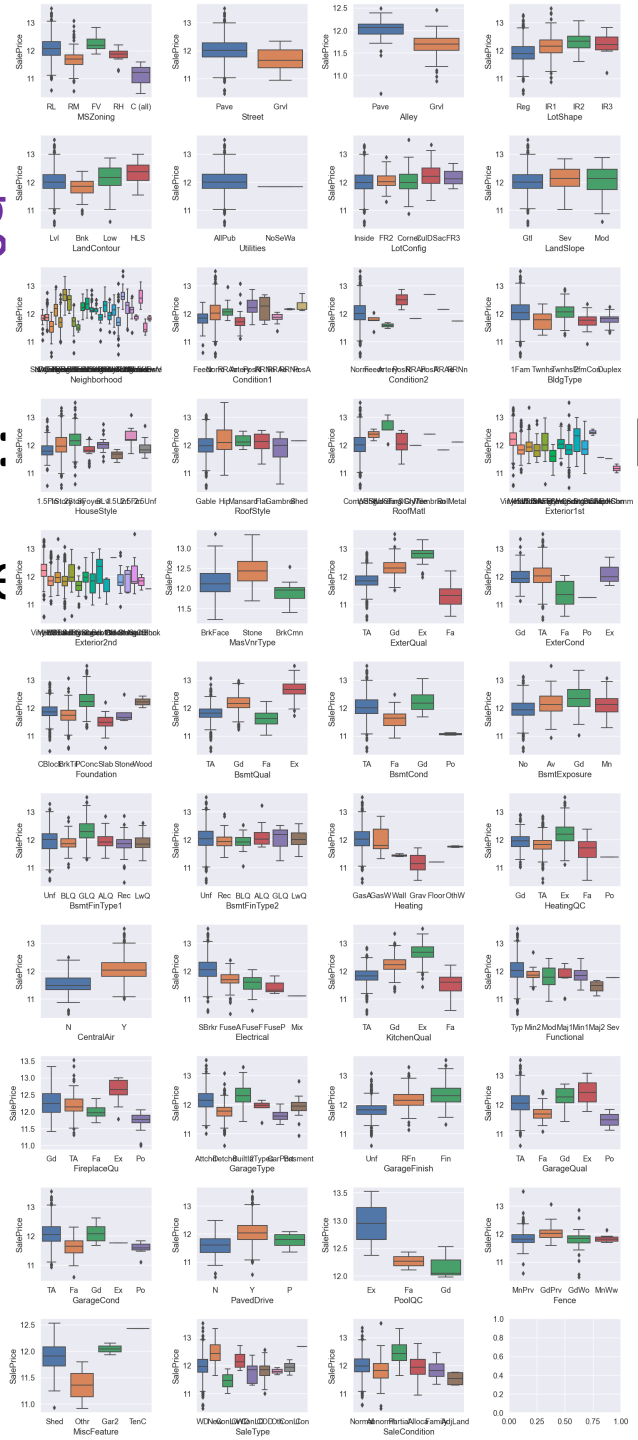


Feature Engineering

- For categorical features: Draw the box plot of the feature and the salesprice to analyze the feature.

Feature Engineering

• For categorical features: … f the feature and the salesprice to analyze the …

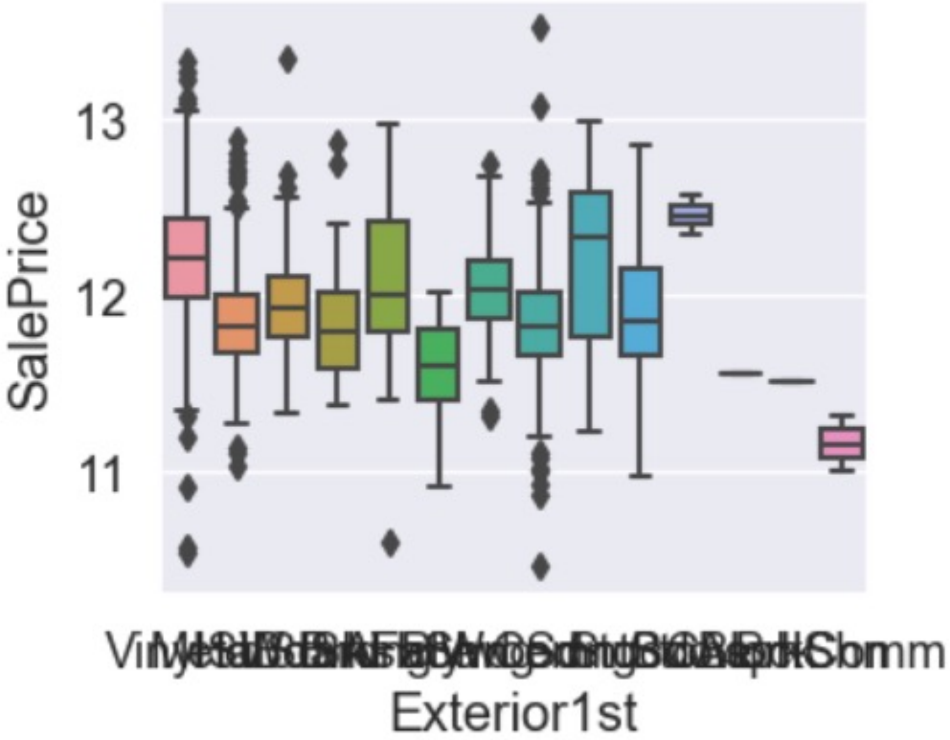


Feature Engineering

- These graphs are harder to read than the scatter plots for the numerical data.
- As the number of features grows, the amount of data we need to accurately be able to distinguish between these features (to give us a prediction) and generalize our model (learned function) grows EXPONENTIALLY.

Feature Engineering

- These graphs are h … or the numerical data.



- As the number of f … e need to accurately be able … s (to give us a prediction) and ge … grows EXPONENTIALLY.

- And it is also hard to extract numerical features. So DROP THEM!

Feature Engineering

- For the categorical features that represents the condition or the quality, or more generalized, ordered features:
- Encode them with numeric values:

```
• order_dict = {"NA" : 0, "Po" : 1, "Fa" : 2, "TA" : 3, "Gd" : 4, "Ex" : 5}
• for feature in order_features:
      data[feature] = data[feature].transform(lambda x: order_dict[x])
```

Feature Engineering

- For other categorical features, just simply use one_hot encode.

```
data = pd.get_dummies(data=data).reset_index(drop=True)
data.shape
```

[30] ✓ 0.0s Python

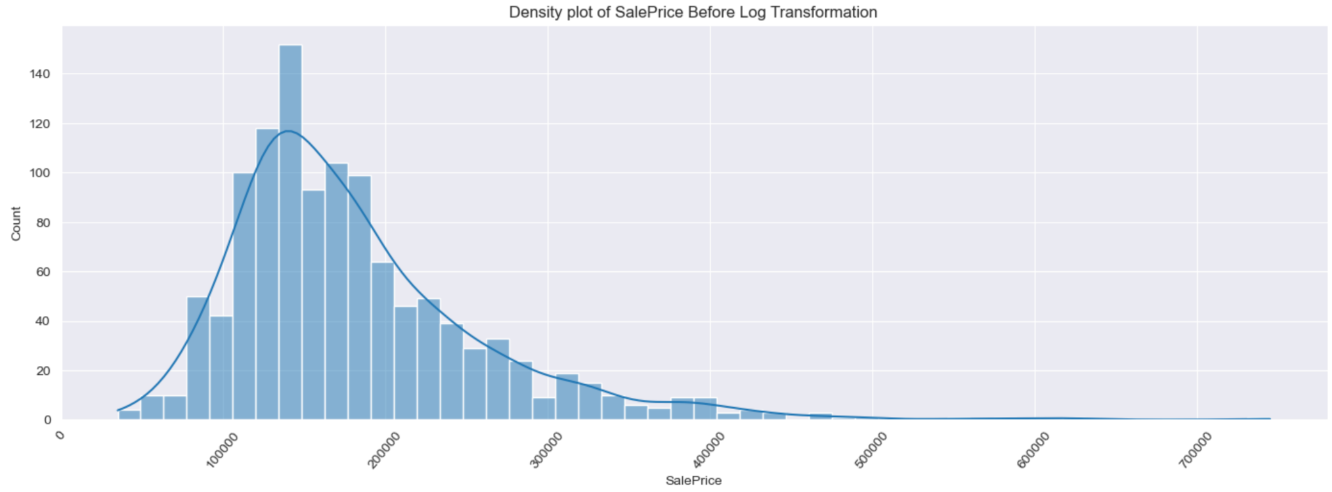
```
(1460, 249)
```

Feature Engineering

- For the value that need to be predicted: SalesPrice.

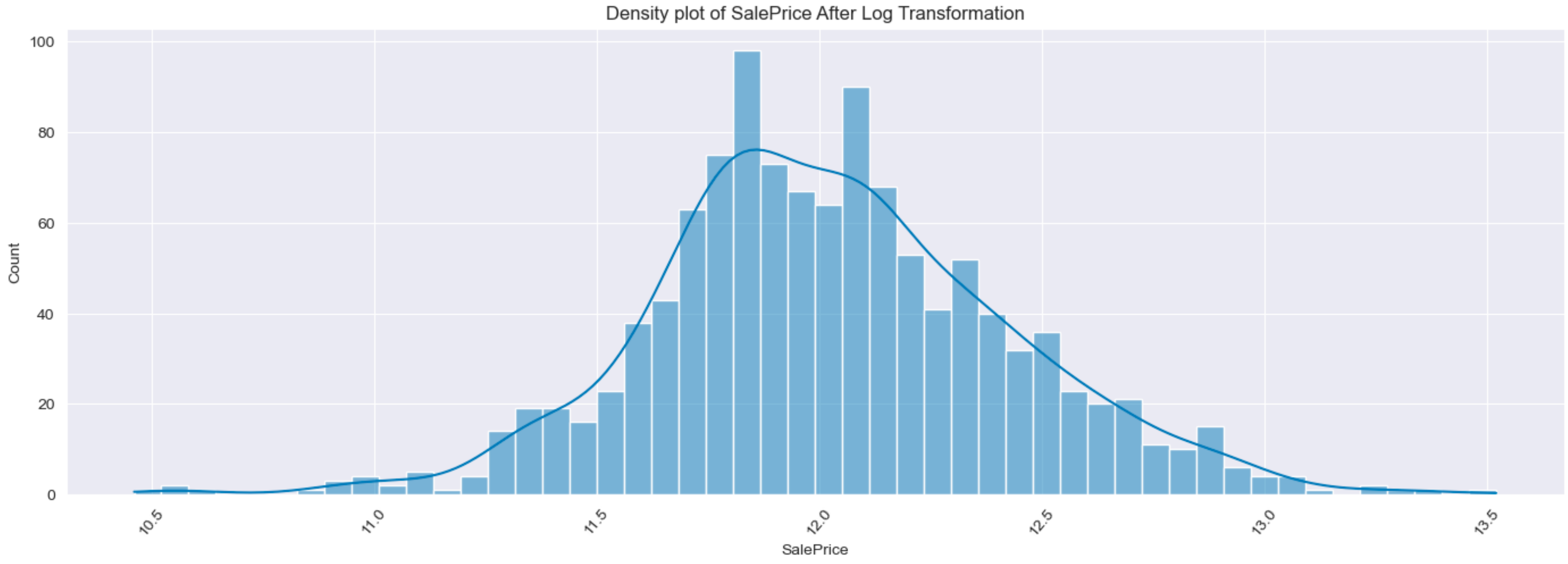
```python
# The Density Plot of SalePrice
plt.figure(figsize=(14, 5))
sns.set_style('darkgrid')
sns.histplot(data=train, x='SalePrice', bins=50, kde=True)
plt.title("Density plot of SalePrice Before Log Transformation")
plt.tight_layout()
plt.xticks(rotation=50)
plt.show()
```

✓ 0.2s Python



Feature Engineering

- For the value that need to be predicted: log1p of SalesPrice.



Modeling: Linear Model

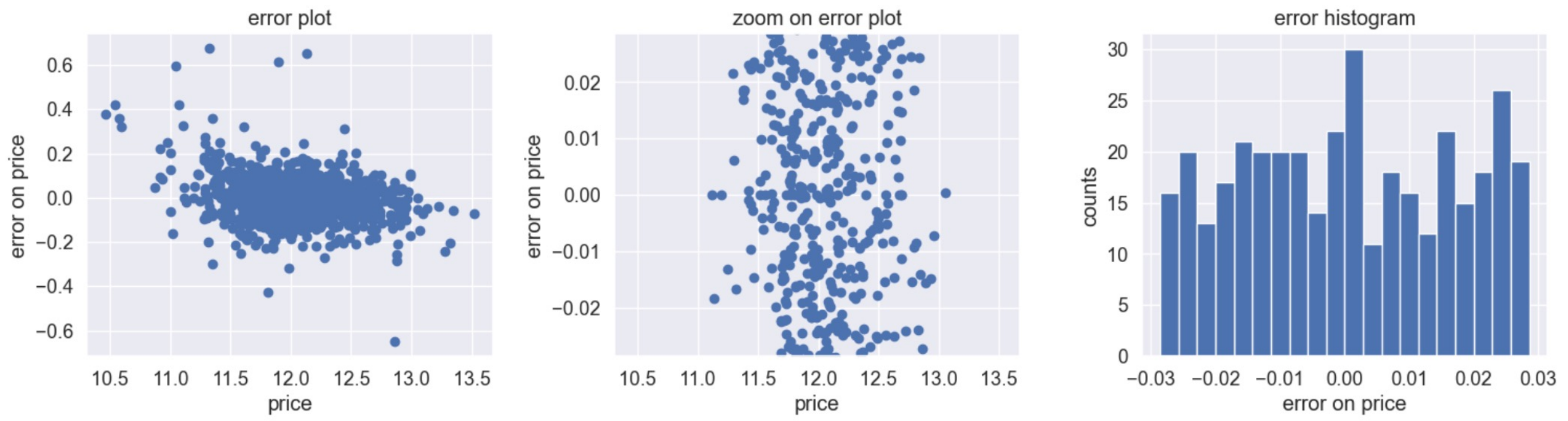
```python
linear_model = LinearRegression()
linear_model.fit(X=x_train, y=y_train)
y_train_pred = linear_model.predict(X=x_train)
mse_train = round(mean_squared_error(y_train_pred, y_train), 5)

print('MSE for Linear Regression is:', mse_train)
```

✓ 0.0s Python

```
MSE for Linear Regression is: 0.00953
```

```python
plot_predict_error(y_pred=y_train_pred, y=y_train)
```

✓ 0.4s Python



Modeling: Lasso

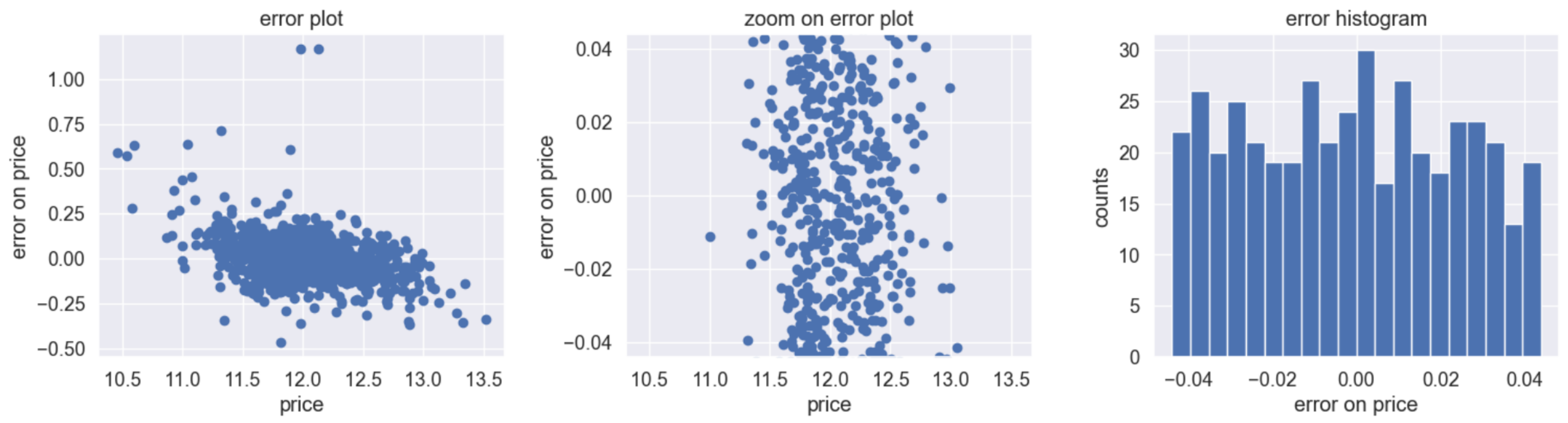
```python
lasso_model = Lasso(alpha=0.001)
lasso_model.fit(X=x_train, y=y_train)
y_train_pred = lasso_model.predict(X=x_train)
mse_train = round(mean_squared_error(y_train_pred, y_train), 5)

print('MSE for Linear Regression is:', mse_train)
```

✓ 0.0s Python

```
MSE for Linear Regression is: 0.01466
```

```python
plot_predict_error(y_pred=y_train_pred, y=y_train)
```

✓ 0.3s Python



Modeling: SVR

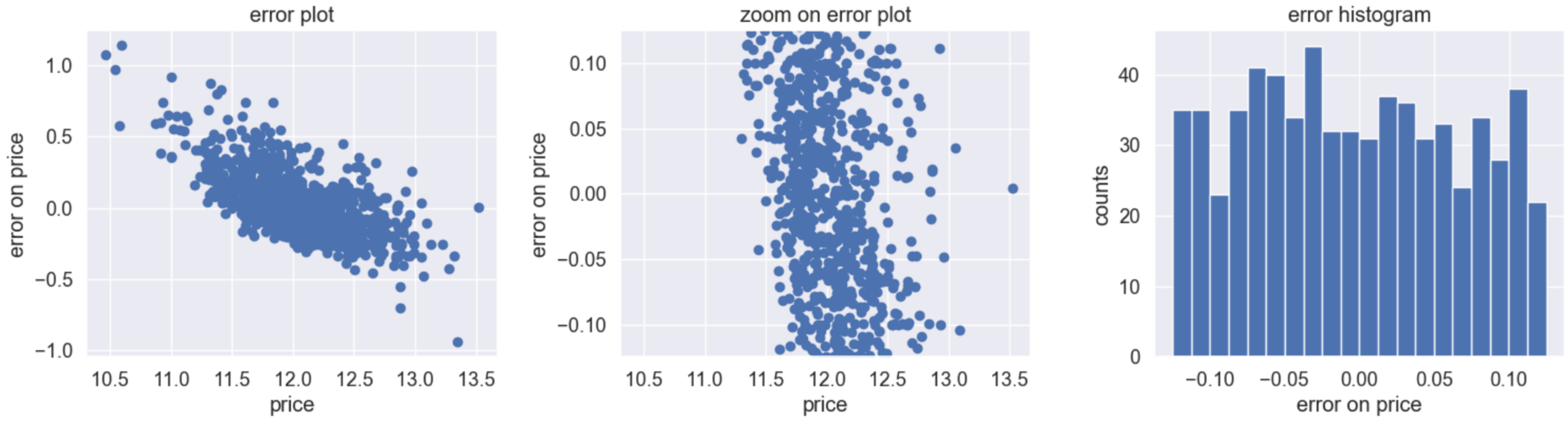
```python
svr_model = SVR()
svr_model.fit(X=x_train, y=y_train)
y_train_pred = svr_model.predict(X=x_train)
mse_train = round(mean_squared_error(y_train_pred, y_train), 5)

print('MSE for Linear Regression is:', mse_train)
```

✓ 0.2s Python

```
MSE for Linear Regression is: 0.04158
```

```python
plot_predict_error(y_pred=y_train_pred, y=y_train)
```

✓ 0.3s Python



Modeling: Random Forest

```python
rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X=x_train, y=y_train)
y_train_pred = rf_model.predict(X=x_train)
mse_train = round(mean_squared_error(y_train_pred, y_train), 5)

print('MSE for Random Forest Regression is:', mse_train)
```
✓ 1.9s Python

```
MSE for Random Forest Regression is: 0.00317
```

```python
plot_predict_error(y_pred=y_train_pred, y=y_train)
```
✓ 0.3s Python

Modeling: GBDT

```python
gbdt_model = GradientBoostingRegressor(n_estimators=3400, max_features=13, max_depth=5,
                                       learning_rate=0.01, subsample=0.8)
gbdt_model.fit(X=x_train, y=y_train)
y_train_pred = gbdt_model.predict(X=x_train)
mse_train = round(mean_squared_error(y_train_pred, y_train), 5)

print('MSE for Random GBDT is:', mse_train)
```

✓ 2.8s Python

```
MSE for Random GBDT is: 0.00037
```

```python
plot_predict_error(y_pred=y_train_pred, y=y_train)
```

✓ 0.3s Python

Modeling: XGB

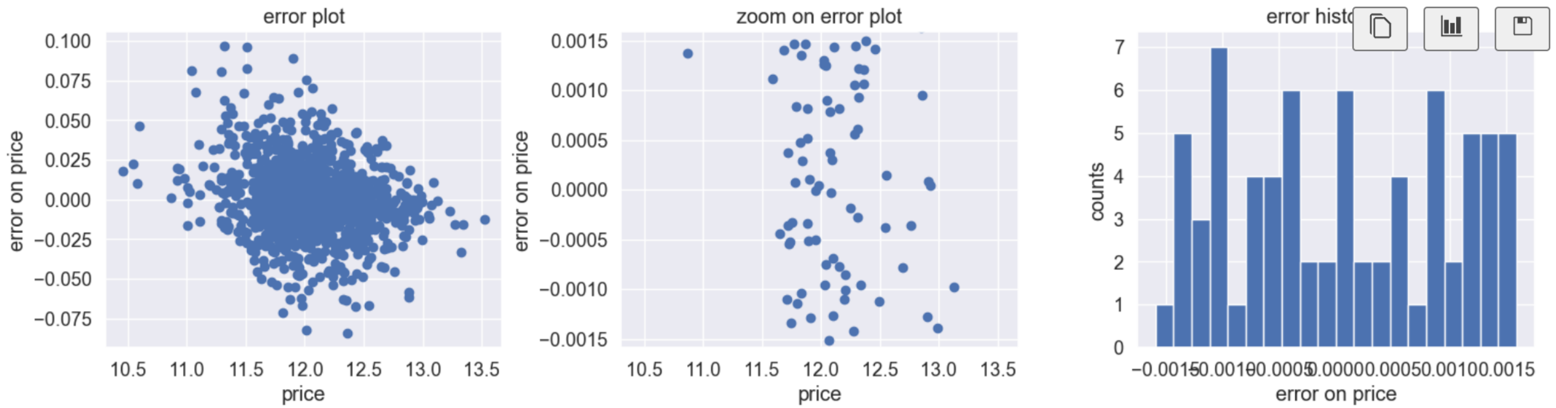
```python
xgb_model = XGBRegressor(n_estimators=2500, max_depth=5, learning_rate=0.01,
                         subsample=0.8, colsample_bytree=0.45)
xgb_model.fit(X=x_train, y=y_train)
y_train_pred = xgb_model.predict(X=x_train)
mse_train = round(mean_squared_error(y_train_pred, y_train), 5)

print('MSE for Random XGB is:', mse_train)
```

✓ 5.1s Python

```
MSE for Random XGB is: 0.00053
```

```python
plot_predict_error(y_pred=y_train_pred, y=y_train)
```

✓ 0.3s Python



Modeling: Stacking

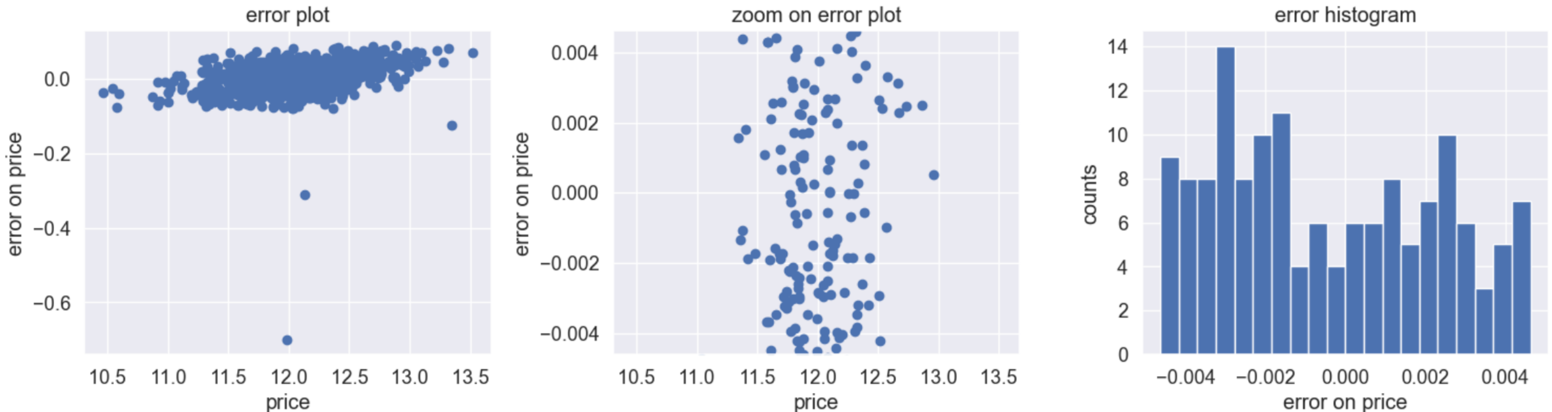
```python
estimators = [
    ('svr', SVR()),
    ('linear', LinearRegression()),
    ('lasso', Lasso()),
    ('rf', RandomForestRegressor(n_estimators=100)),
    ('gbdt', GradientBoostingRegressor(n_estimators=3400, max_features=13, max_depth=5, learning_rate=0.01, subsample
    ('xgb', XGBRegressor(n_estimators=2500, max_depth=5, learning_rate=0.01, subsample=0.8, colsample_bytree=0.45))
]

stacking_model = StackingRegressor(estimators=estimators)
stacking_model.fit(X=x_train, y=y_train)
y_train_pred = stacking_model.predict(X=x_train)
mse_train = round(mean_squared_error(y_train_pred, y_train), 5)

print('MSE for stacking model is:', mse_train)
```

✓ 51.2s Python

```
MSE for stacking model is: 0.00154
```

```python
plot_predict_error(y_pred=y_train_pred, y=y_train)
```

✓ 0.3s Python



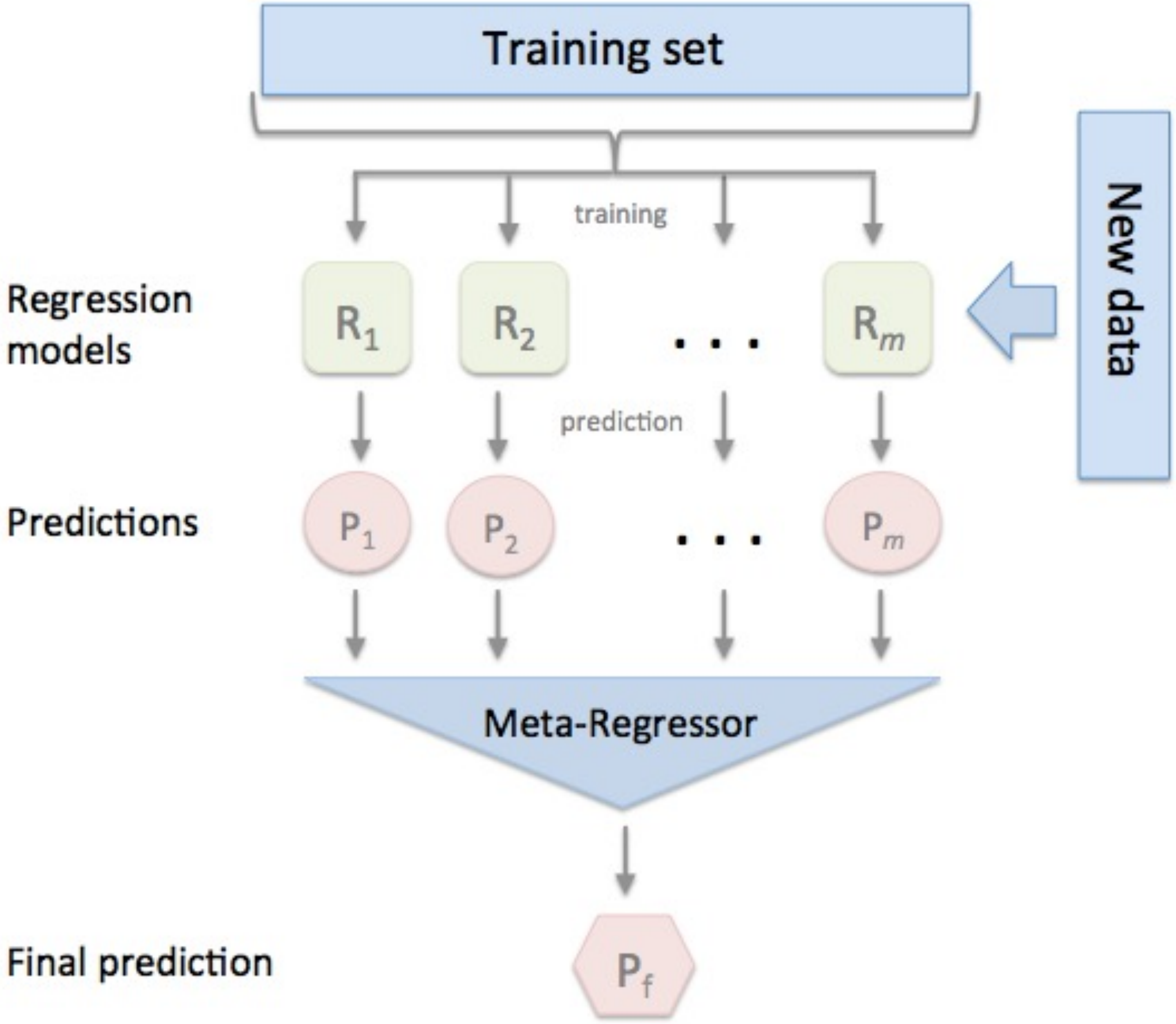
Performance On Test Set

```python
model_set = [
    ('linear', linear_model), ('lasso', lasso_model), ('svr', svr_model),
    ('RF', rf_model), ('GBDT', gbdt_model), ('XGB', xgb_model),
    ('stacking', stacking_model)
]

for name, model in model_set:
    y_test_pred = model.predict(X=x_test)
    mse_test = round(mean_squared_error(y_test_pred, y_test), 5)
    print('The mse on the test set of model', name, 'is: \t', mse_test)
```

✓ 0.2s Python

```
The mse on the test set of model linear is:      0.01287
The mse on the test set of model lasso is:       0.01315
The mse on the test set of model svr is:         0.0372
The mse on the test set of model RF is:          0.0151
The mse on the test set of model GBDT is:        0.01097
The mse on the test set of model XGB is:         0.01147
The mse on the test set of model stacking is:    0.00999
```

Why Stacking?



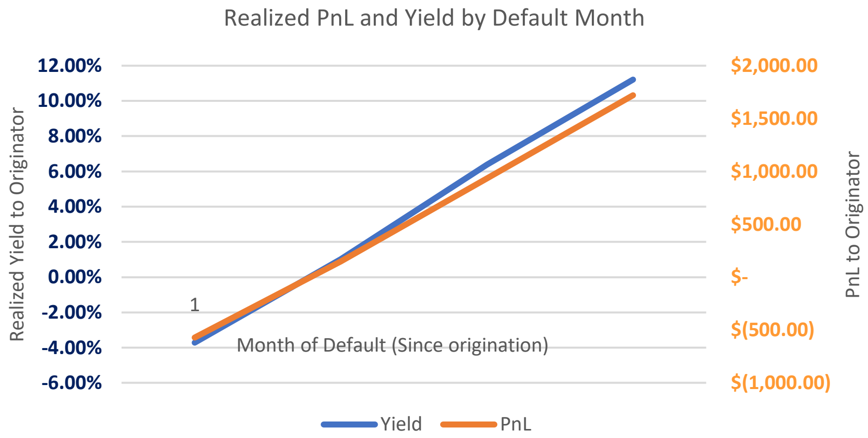
Thank you!

Predicting the Timing of Consumer Loan Defaults

Ziming Huang
Project ID: 2

Why WHEN vs IF is important

- **Context**:
  - Marketplace lending and consumer loans is a $10bb-per-year business for banks and fintech companies
  - In order to be profitable, the originators of the loans need to accurately model default risk of the underlying borrowers
  - LendingClub is an online peer-to-peer lending platform that connects borrowers with individual and institutional investors who are willing to fund their loans; LendingClub essentially acts as an intermediary, connecting borrowers seeking loans with investors looking to earn returns by lending money
- **Problem:**
  - Traditional credit models often do a good job capturing IF a borrower defaults; typically a simple logistic regression will achieve high accuracy
  - However, the simple approach does not predict WHEN a borrower defaults -> the originator / investor realizes different PnL depending on WHEN the default happens
- **Significance of the problem:**
  - In our hypothetical example below, we assume:
    - 100 borrowers take out $100 loan each
    - 15 borrowers default, and the other 85 borrowers pay in full according to their contractual schedule
    - Our model predicts the defaulters with 100% accuracy, but does not predict WHEN the defaults happen
    - Along the x-axis we model scenarios in which the defaults all happen in month 1, 2, …
  - We can see that the PnL realized to the originator / investor is different depending on WHEN the default happens, even though the total # of defaults is held unchanged



Default Month	% Borrowers Defaulted	Yield	PnL
1	20%	-3.71%	-571.25
6	20%	-1.63%	-243.86
12	20%	1.02%	149.00
18	20%	3.72%	541.87
24	20%	6.38%	934.73
30	20%	8.90%	1,327.60
31	20%	9.30%	1,393.07
36	20%	11.21%	1,720.46

Mathematical Framework

For each loan, denoted by index l, we define the following:

x_l: loan-level information of loan l such as FICO or DTI

$S_{l,t}$: status (0: non-default, 1: defaulted) of loan l at time t.

τ_l: timing of default of loan l; note here if loan l pays in full, $\tau_l = \infty$.

T_l: original loan term of loan l; this is a known constant for each loan.

Dataset: $X = \{x_1, x_2, ..., x_N\} \in R^{N*d}$: loan-level information such as FICO, DTI; this is our independent variable dataset. $Y = \{y_1, y_2, ..., y_N\} \in R^N$: default time of each loan.

We model each loan l as follows:

when $t <= T_l$

$P(S_{l,t} = 0|S_{l,t-1} = 0) = \rho(x_l, t)$

$P(S_{l,t} = 1|S_{l,t-1} = 0) = 1 - \rho(x_l, t)$

$P(S_{l,t} = 0|S_{l,t-1} = 1) = 0$

$P(S_{l,t} = 0|S_{l,t-1} = 1) = 1$

Transition probability

when $t > T_l$

$P(S_{l,t} = 0|S_{l,t-1} = 0) = 1$

$P(S_{l,t} = 1|S_{l,t-1} = 0) = 0$

$P(S_{l,t} = 0|S_{l,t-1} = 1) = 0$

$P(S_{l,t} = 0|S_{l,t-1} = 1) = 1$

The definition for when $t > T_l$ is needed so that the probabilities sum up to 1.

Here, we assume the transition probability is dependent on time and loan-level information. For notation, we say the function ρ is parameterized by a set if parameters μ.

$P(\tau_l = t) = P(S_0 = 0, S_1 = 0, ..., S_t = 1)$

$= P(S_t = 1|S_{t-1} = 0, ..., S_0 = 0)P(S_{t-1} = 0|S_{t-2} = 0, ..., S_0 = 0)...P(S_1 = 0|S_0 = 0)P(S_0 = 0)$

$= (1 - \rho(x_l, t)) \prod_{i=1}^{t-1} \rho(x_l, i)$

(when $t > T_l$)

$P(\tau_l = t) = 0$

Lastly, $\tau_l = \infty$ if and only if loan never defaults, i.e., $S_0 = 0, S_1 = 0, ..., S_{T_l} = 0$. Hence:

$P(\tau_l = \infty) = P(S_0 = 0, S_1 = 0, ..., S_{T_l} = 0) = \prod_{i=1}^{T_l} \rho(x_l, i)$

Distribution of default time

The distribution function of τ_l can be rewritten as:

$f_{\tau_l}(t) = I\{t < \infty\}(1 - \rho(x_l, t)) \prod_{i=1}^{t-1} \rho(x_l, i) + I\{t = \infty\} \prod_{i=1}^{T_l} \rho(x_l, i)$

Let $\delta_l = I\{t < \infty\} = I\{t <= T_l\}$ (note that this δ_l denotes whether loan l defaults, then:

$f_{\tau_l}(t) = ((1 - \rho(x_l, t)) \prod_{i=1}^{t-1} \rho(x_l, i))^{\delta_l} (\prod_{i=1}^{T_l} \rho(x_l, i))^{(1-\delta_l)}$

$ln(f_{\tau_l}(t)) = \delta_l * \{ln(1 - \rho(x_l, t)) + \sum_{i=1}^{t-1} ln(\rho(x_l, i))\} + (1 - \delta_l) * \sum_{i=1}^{T_l} ln(\rho(x_l, i))$

The likelihood of the set of outcome Y, given parameter μ, can be expressed as follows:

$P(Y|\mu, X) = \prod_{l=1}^{N} f_{\tau_l}(y_l)$

$= \prod_{l=1}^{N} \{I\{t < \infty\}(1 - \rho(x_l, t)) \prod_{i=1}^{t-1} \rho(x_l, i) + I\{t = \infty\} \prod_{i=1}^{T_l} \rho(x_l, i)\}$

$lnP = \sum_{l=1}^{N} ln(f_{\tau_l}(y_l))$

$= \sum_{i=1}^{N} \delta_l * \{ln(1 - \rho(x_l, t)) + \sum_{i=1}^{t-1} ln(\rho(x_l, i))\} + (1 - \delta_l) * \sum_{i=1}^{T_l} ln(\rho(x_l, i))$

Log likelihood

The negative log-likelihood can be expressed as:

$NLL = -lnP$

Implementation and Results (preliminary)

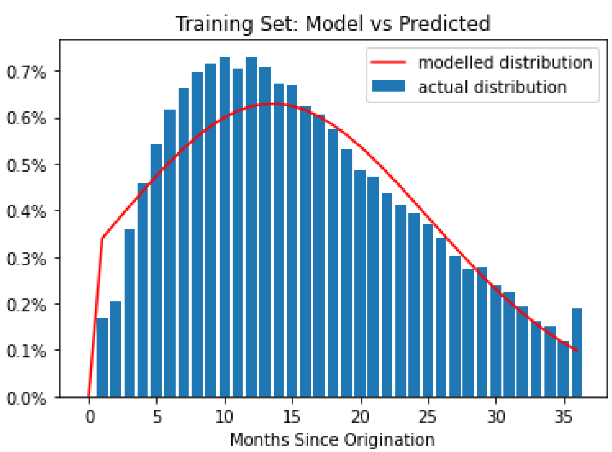
- **Implementation**:
- For simplicity, we only use FICO for the loan-level independent variable. We then parameterize ρ as follows:

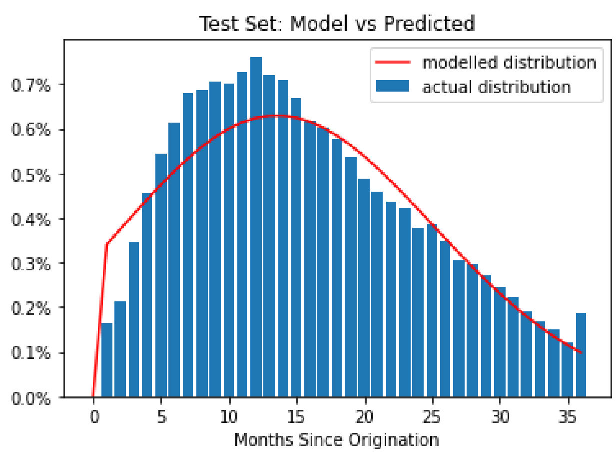
$$\rho(x_l, t) = \frac{1}{1+exp(-(\beta_F x_l + \beta_{F,t2} x_l t^2 + \beta_{F,t} x_l t + at^2 + bt + c))}$$

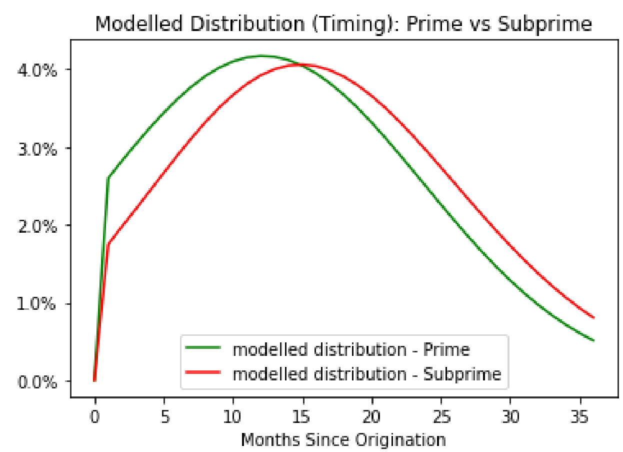
  - \- > i.e., the transition probability is linear in FICO and quadratic in time, and the cross products, FICO x time and FICO x time-squared, are also included to capture any non-linear relationship between the two variables.

- **Solution**:

beta_fico	beta_fico_time_sqaured	beta_fico_time	t squared	t	const
0.00831	0.00000	0.00028	0.00428	-0.30454	-0.00996







Binary Image Classifier on Smaller Datasets

Dec 12, 2023

Jiaming Li

Smaller Datasets

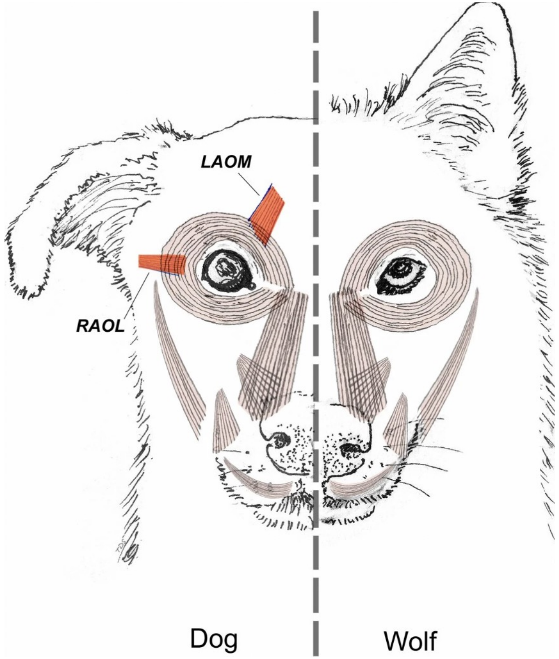
- Each class has about 300 images

- Both scaled to size 256*256 for training.

SVM v.s. CNN-SVM v.s. CNN

- Algorithm Complexity:
  - SVM/ CNN-SVM are simpler algorithms which are suitable for smaller datasets to prevent
- Data Complexity:
  - Pure CNN can do a better job finding patterns/ capturing complex features.
- Runtime/Computation Complexity:
  - Training CNN-SVM/CNN can be time consuming.

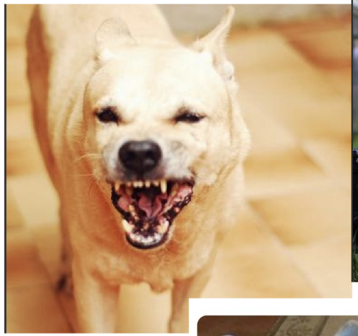
Now the algorithms have been decided...

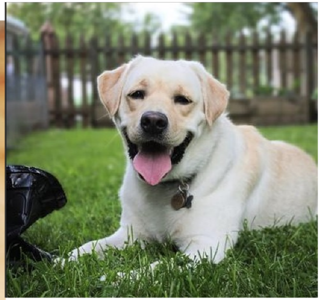


- Dataset:

  guilty/not guilty

  (300 each)





Picture from: Juliane Kaminski, Bridget M. Waller, Rui Diogo, Adam Hartstone-Rose, and Anne M. Burrows. Evolution of facial muscle anatomy in dogs. Proceedings of the National Academy of Sciences, 116(29):14677–14681, 2019.

Progress

- Data split: 25% test, 75% train
- SVM seems to work just fine…
  - Each image is converted into a array of length 196608 (256*256*3)
  - Without regulation/data augmentation (accuracy: 0.76)

Progress

- Data split: 10% test, 20% validation, 70% train
- CNN_SVM & CNN work better:

- With the same structure of CNN and number of epochs (10):

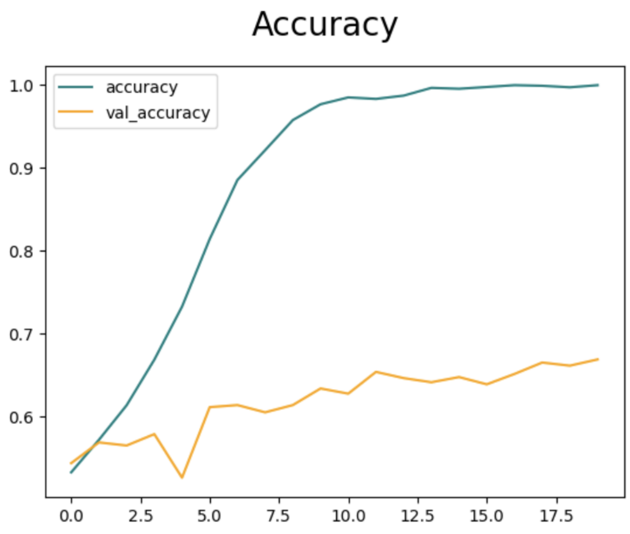
  CNN has better performance than CNN-SVM

```
Precision:        0.875
Recall:           0.6774193644523621
Accuracy:         0.7796609997749329
```

```
Precision:        0.9354838728904724
Recall:           0.7837837934494019
Accuracy:         0.8305084705352783
```

Preventing CNN From Overfitting

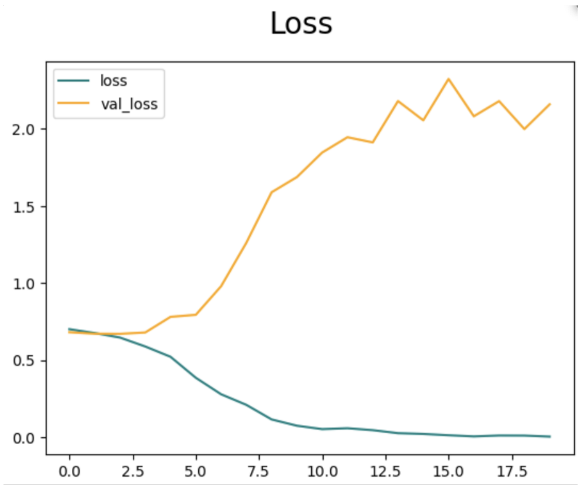
- CNN in comparison to SVM, is very likely to overfit.
- Adding more convolution layers (with fewer number of filters) helped reducing the total number of trainable parameters.
- Add Dropout() layers as regulation.

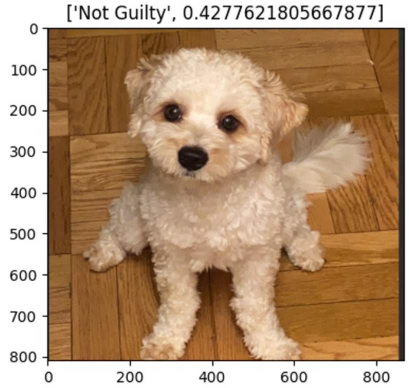
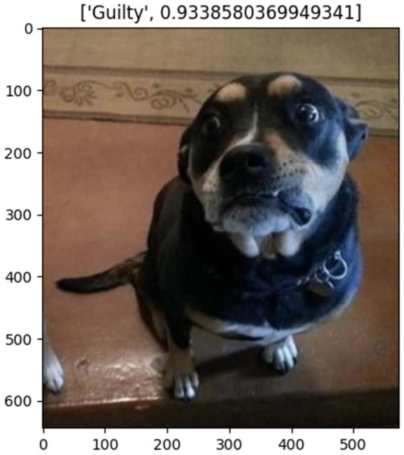
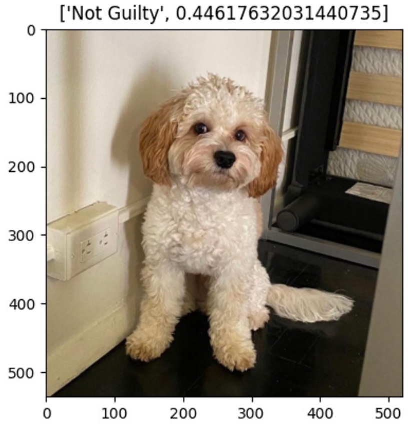
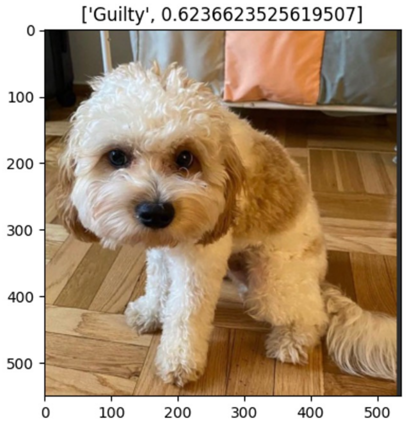
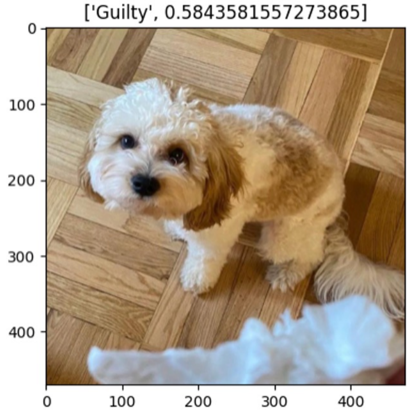
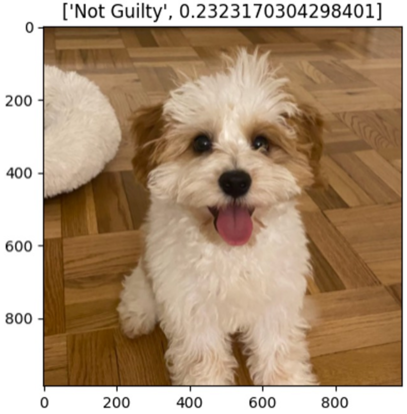
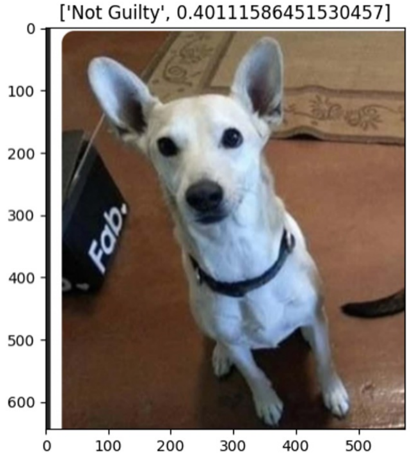
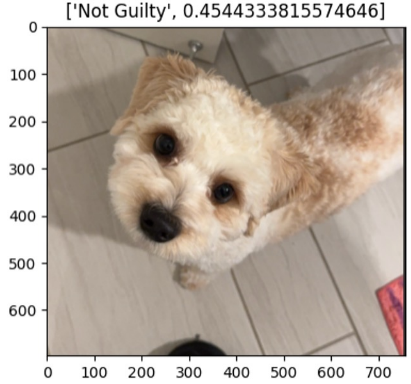


Things To Add On

- When dealing with larger dataset (2k+ samples)...
  - Changing / adding layers may not be enough

- K-fold validation/data augmentation ?




['Not Guilty', 0.4277621805667877]

['Guilty', 0.9338580369949341]

['Not Guilty', 0.44617632031440735]

['Guilty', 0.6236623525619507]

['Guilty', 0.5843581557273865]

['Not Guilty', 0.2323170304298401]

['Not Guilty', 0.40111586451530457]

['Not Guilty', 0.4544333815574646]

Predicting Online Review Helpfulness: From Linear Models to Transformers

Mengzhu Chen

Project ID: 4

Introduction

In today's digital marketplace, online reviews significantly influence consumer purchase decisions. This project aims to predict the helpfulness of Amazon product reviews using a range of ML and DL models:

- Linear Regression (LR)
- Support Vector Machines (SVM)
- Decision Trees (DT)
- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM) networks
- Transformer

Amazon Review Dataset

Sample review data:


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


Focus on Home & Kitchen category

- 24,646 rewiews with at least 15 votes
- 80% as training set, 20% as test set

Two values in helpful field

- Number of upvotes: 2
- Total number of votes: 3 (1 downvote)

Two predicting targets

- Upvote ratio: 2/3
- Number of upvotes: 2
  - Follows long-tail distribution, use log(2)
  - To prevent log(0), use log(2+1)

Feature Comparison

Model	Feature	RMSE for upvote ratio	RMSE for log number of upvotes
Linear Regression	Count	0.1871	0.9653
Linear Regression	TF-IDF	0.1866	0.9690
SVM	Count	0.2215	0.9744
SVM	TF-IDF	0.1883	0.9674
Decision Trees	Count	0.1899	0.9650
Decision Trees	TF-IDF	0.1912	0.9655

Model Comparison

Model	Feature	RMSE for upvote ratio	RMSE for log number of upvotes
Linear Regression	TF-IDF	0.1866	0.9690
SVM	TF-IDF	0.1883	0.9744
Decision Trees	TF-IDF	0.1912	0.9655
RNN	Word2Vec	0.1711	0.8822
LSTM	Word2Vec	0.1710	0.8818
Transformer	BERT	**0.1473**	**0.8415**

Feature Fusion

Feature Fusion on Transformer	RMSE for upvote ratio	RMSE for log number of upvotes
No fusion	**0.1473**	0.8415
Concatenate length number to feature vector	0.1579	**0.8331**
Concatenate rating star number to feature vector	0.1600	0.8528
Concatenate “star x” text before input text	0.1483	0.8750
Concatenate summary text before input text	0.1609	0.8681

Conclusion

- TF-IDF/word count make little difference for ML models in this problem
- DL models perform better than traditional ML models for this task
- Pretrained large-scale Transformer performs better than RNN/LSTM
- Fusing metadata may mislead the Transformer model in this problem

Predict Student Performance from Game Play

Xuanbing Zhu
Project ID: 5

Jo Wilder online educational game

Enhance educational game design

How?

For example, if a student is keeping getting wrong answers during the game, then the game should give this student easier questions, so that he could continue to play the game and not feeling defeated.

Dataset

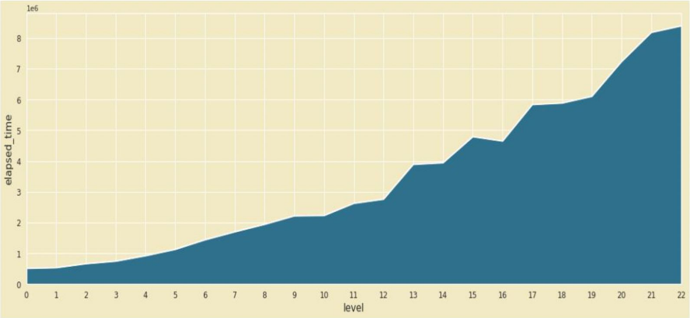
Columns

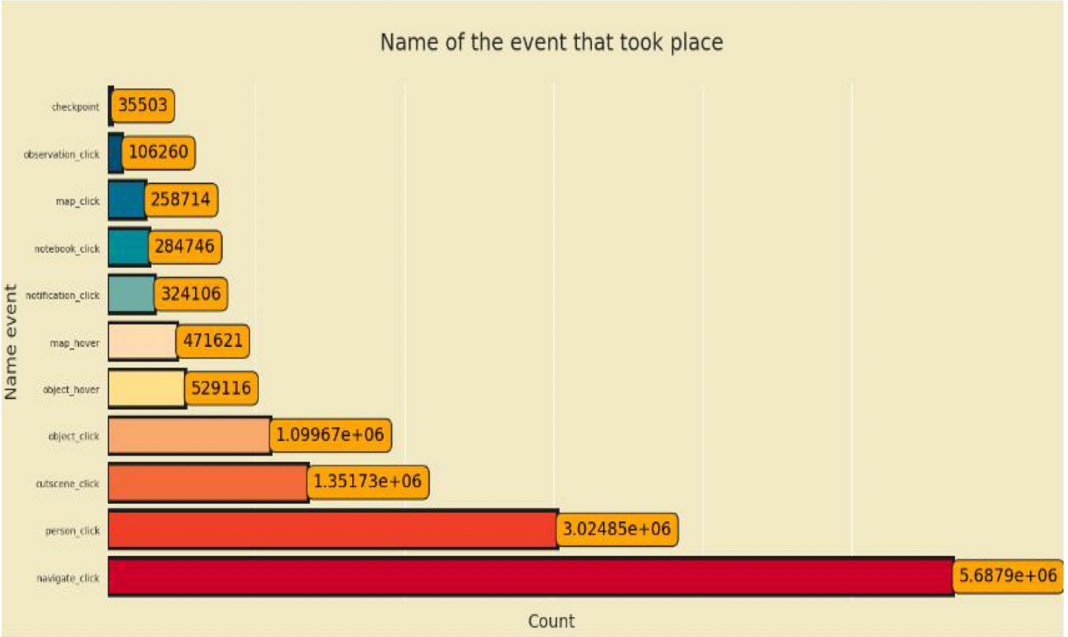
- **session_id** - the ID of the session the event took place in
- **index** - the index of the event for the session
- **elapsed_time** - how much time has passed (in milliseconds) between the start of the session and when the event was recorded
- **event_name** - the name of the event type
- **name** - the event name (e.g. identifies whether a notebook_click is is opening or closing the notebook)
- **level** - what level of the game the event occurred in (0 to 22)
- **page** - the page number of the event (only for notebook-related events)
- **room_coor_x** - the coordinates of the click in reference to the in-game room (only for click events)
- **room_coor_y** - the coordinates of the click in reference to the in-game room (only for click events)
- **screen_coor_x** - the coordinates of the click in reference to the player's screen (only for click events)
- **screen_coor_y** - the coordinates of the click in reference to the player's screen (only for click events)
- **hover_duration** - how long (in milliseconds) the hover happened for (only for hover events)
- **text** - the text the player sees during this event
- **fqid** - the fully qualified ID of the event
- **room_fqid** - the fully qualified ID of the room the event took place in
- **text_fqid** - the fully qualified ID of the
- **fullscreen** - whether the player is in fullscreen mode
- **hq** - whether the game is in high-quality
- **music** - whether the game music is on or off
- **level_group** - which group of levels - and group of questions - this row belongs to (0-4, 5-12, 13-22)

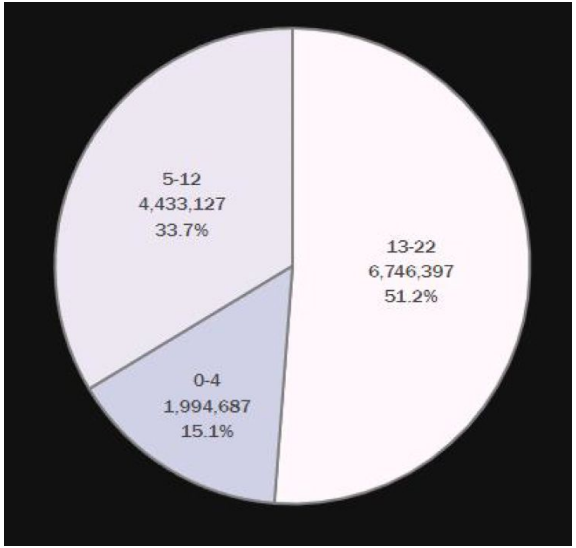
	session_id	index	elapsed_time	event_name	name	level	page	room_coor_x	room_coor_y	screen_coor_x	room_coor_y	screen_coor_x	scre
0	20090312431273200	0	0	cutscene_click	basic	0	NaN	-413.991405	-159.314686	380.0	-159.314686	380.0	
1	20090312431273200	1	1323	person_click	basic	0	NaN	-413.991405	-159.314686	380.0	-159.314686	380.0	
2	20090312431273200	2	831	person_click	basic	0	NaN	-413.991405	-159.314686	380.0	-159.314686	380.0	
3	20090312431273200	3	1147	person_click	basic	0	NaN	-413.991405	-159.314686	380.0	-159.314686	380.0	
4	20090312431273200	4	1863	person_click	basic	0	NaN	-412.991405	-159.314686	381.0	-159.314686	381.0	

hover_duration	text	fqid	room_fqid	text_fqid	fullscreen	hq	music	level_group
NaN	undefined	intro	tunic.historicalsociety.closet	tunic.historicalsociety.closet.intro	0	0	1	0-4
NaN	Whatcha doing over there, Jo?	gramps	tunic.historicalsociety.closet	tunic.historicalsociety.closet.gramps.intro_0_...	0	0	1	0-4
NaN	Just talking to Teddy.	gramps	tunic.historicalsociety.closet	tunic.historicalsociety.closet.gramps.intro_0_...	0	0	1	0-4
NaN	I gotta run to my meeting!	gramps	tunic.historicalsociety.closet	tunic.historicalsociety.closet.gramps.intro_0_...	0	0	1	0-4
NaN	Can I come, Gramps?	gramps	tunic.historicalsociety.closet	tunic.historicalsociety.closet.gramps.intro_0_...	0	0	1	0-4

Data Visualization







Feature Engineering

calculating event durations, grouping variables by session ID, and deriving some time-related features

```
def feature_engineer(x, grp, use_extra, feature_suffix):
    aggs = [
        pl.col("index").count().alias(f"session_number_{feature_suffix}"),

        #text
        *[pl.col('index').filter(pl.col('text').str.contains(c)).count().alias(f'word_{c}') for c in DIALOGS],
        *[pl.col("elapsed_time_diff").filter((pl.col('text').str.contains(c))).mean().alias(f'word_mean_{c}') for c in
          DIALOGS],
        *[pl.col("elapsed_time_diff").filter((pl.col('text').str.contains(c))).std().alias(f'word_std_{c}') for c in
          DIALOGS],
        *[pl.col("elapsed_time_diff").filter((pl.col('text').str.contains(c))).max().alias(f'word_max_{c}') for c in
          DIALOGS],
        *[pl.col("elapsed_time_diff").filter((pl.col('text').str.contains(c))).sum().alias(f'word_sum_{c}') for c in
          DIALOGS],
        *[pl.col("elapsed_time_diff").filter((pl.col('text').str.contains(c))).median().alias(f'word_median_{c}') for c
          in DIALOGS],

        *[pl.col('text_code').filter(pl.col('text_code') == c).count().alias(f'{c}_text_code_counts{feature_suffix}') for c in test_list],
        *[pl.col("elapsed_time_diff").filter((pl.col('text_code') == c)).mean().alias(f'{c}_text_mean_{feature_suffix}') for c in
          test_list],
        *[pl.col("elapsed_time_diff").filter((pl.col('text_code') == c)).std().alias(f'{c}_text_std_{feature_suffix}') for c in
          test_list],
        *[pl.col("elapsed_time_diff").filter((pl.col('text_code') == c)).max().alias(f'{c}_text_max_{feature_suffix}') for c in
          test_list],
        *[pl.col("elapsed_time_diff").filter((pl.col('text_code') == c)).sum().alias(f'{c}_text_sum_{feature_suffix}') for c in
          test_list],
        *[pl.col("elapsed_time_diff").filter((pl.col('text_code') == c)).median().alias(f'{c}_text_median_{feature_suffix}') for c
          in test_list],
```

Model Selection

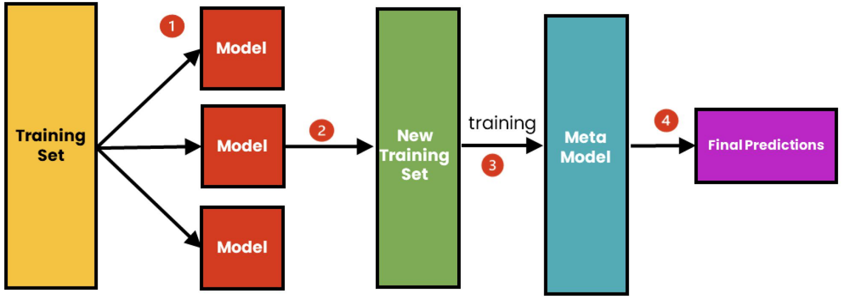
First, we tried **XGBoost**, **LightGBM**, and **CatBoost**

LightGBM performs the best.

Stacking

We used the predictions from our LightGBM models as inputs for a higher-level model, a Logistic Regression in our case, to refine and enhance our predictions.

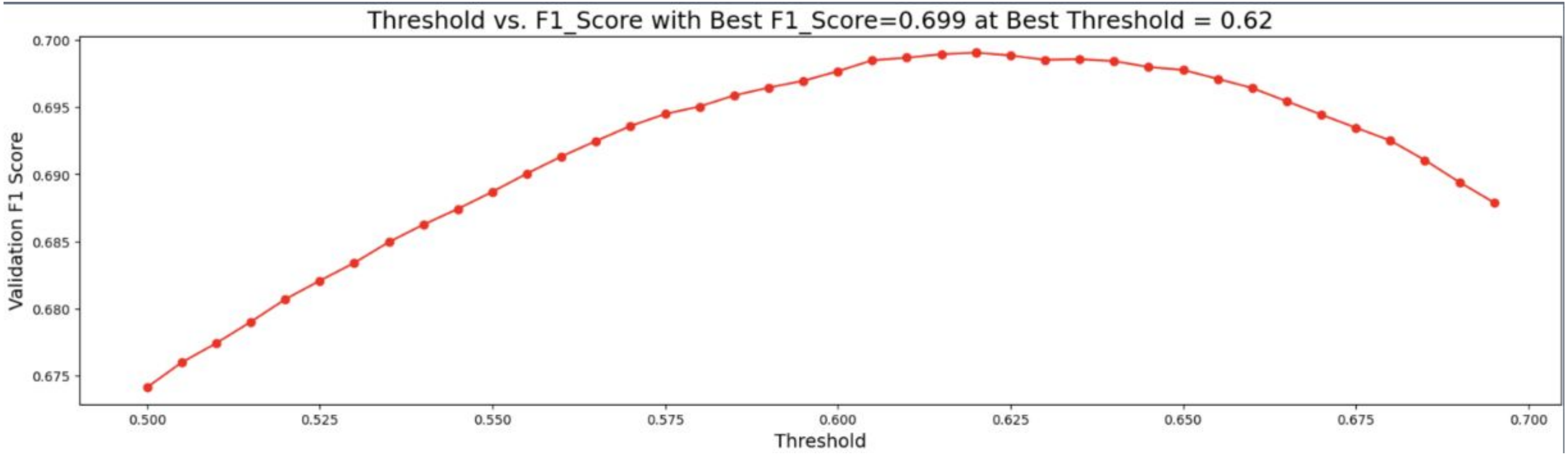
The Process of Stacking




```python
for fold, (train_idx, valid_idx) in enumerate(gkf.split(X=df, groups=df.index)):
    # TRAIN DATA
    train_x = oof_cat.iloc[train_idx]
    train_users = train_x.index.values
    train_y = targets.loc[targets.q == q].set_index('session').loc[train_users]

    # VALID DATA
    valid_x = oof_cat.iloc[valid_idx]
    valid_users = valid_x.index.values
    valid_y = targets.loc[targets.q == q].set_index('session').loc[valid_users]



    lgb_train = lgb.Dataset(train_x[FEATURES].astype('float32'), train_y['correct'].values)
    lgb_eval = lgb.Dataset(valid_x[FEATURES].astype('float32'), valid_y['correct'].values)


    model = LogisticRegression(random_state=0).fit(train_x[FEATURES].astype('float32'), train_y['correct'].values)

    y = valid_y
    y_hat = model.predict_proba(valid_x[FEATURES])[:,1]
    models_stack[(fold, q)] = model

    oof_cat_stack.loc[valid_users,f'meta_{q}'] = y_hat


    results_stack[q - 1][0].append(y)
    results_stack[q - 1][1].append(y_hat)
```

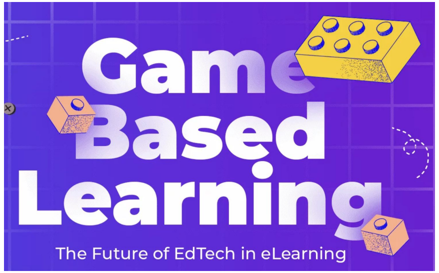
F1 Score

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$



Conclusion


START


Game
Based
Learning
The Future of EdTech in eLearning

Thank you for listening!

Q&A

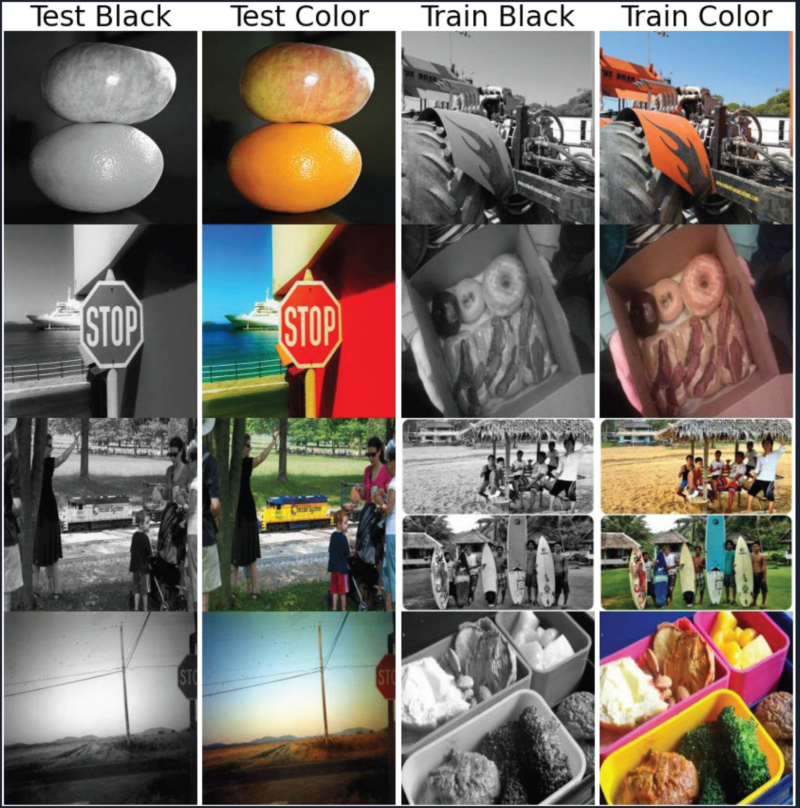
Image Colorization In Machine Learning

Author: Zhou Zhou, Yunqing Zhu
New York University Courant Institute of Mathematical Sciences

Introduction

- We try to add color to grayscale image.
- We use machine learning methods as baseline and us convolution neural network as advance model to implement image colorization

Dataset



Machine Learning Methods

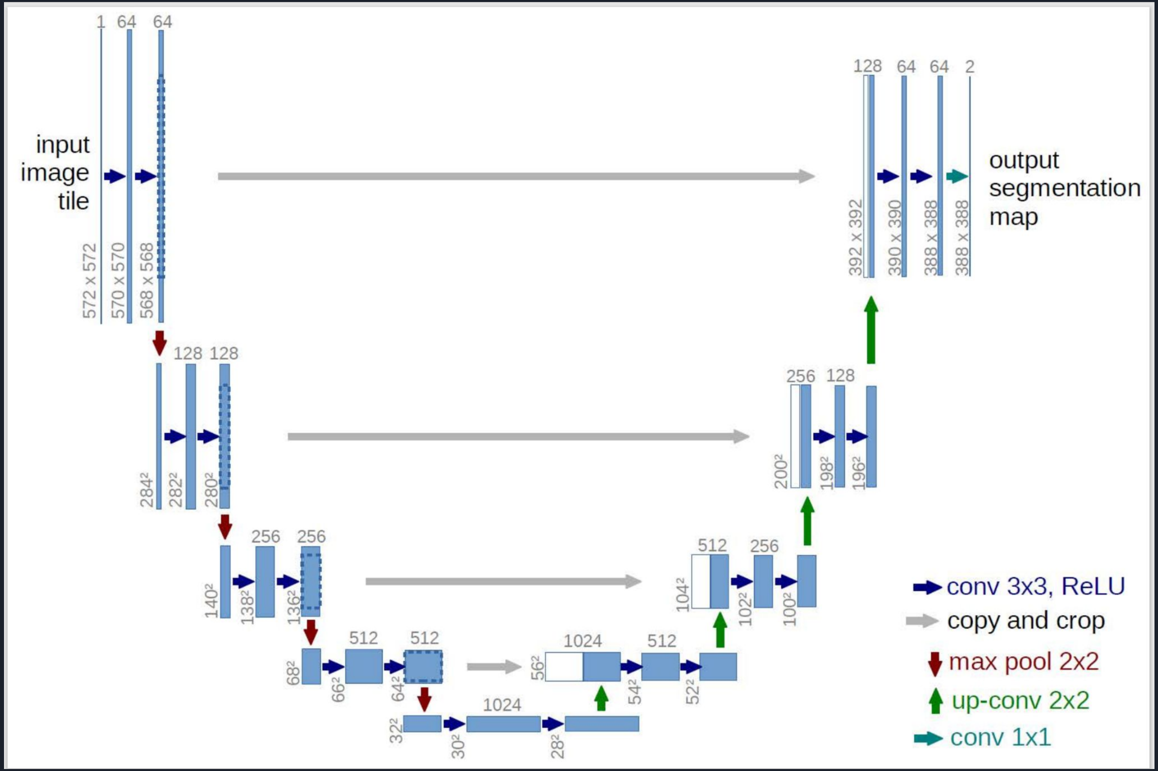
- Linear regression
- Gradient boosting
- Decision Tree
- Random Forest

Machine Learning Methods

L:X_11	L:X_12	L:X_13
L:X_21	A:Y_11	L:X_22
L:X_31	L:X_32	L:X_33

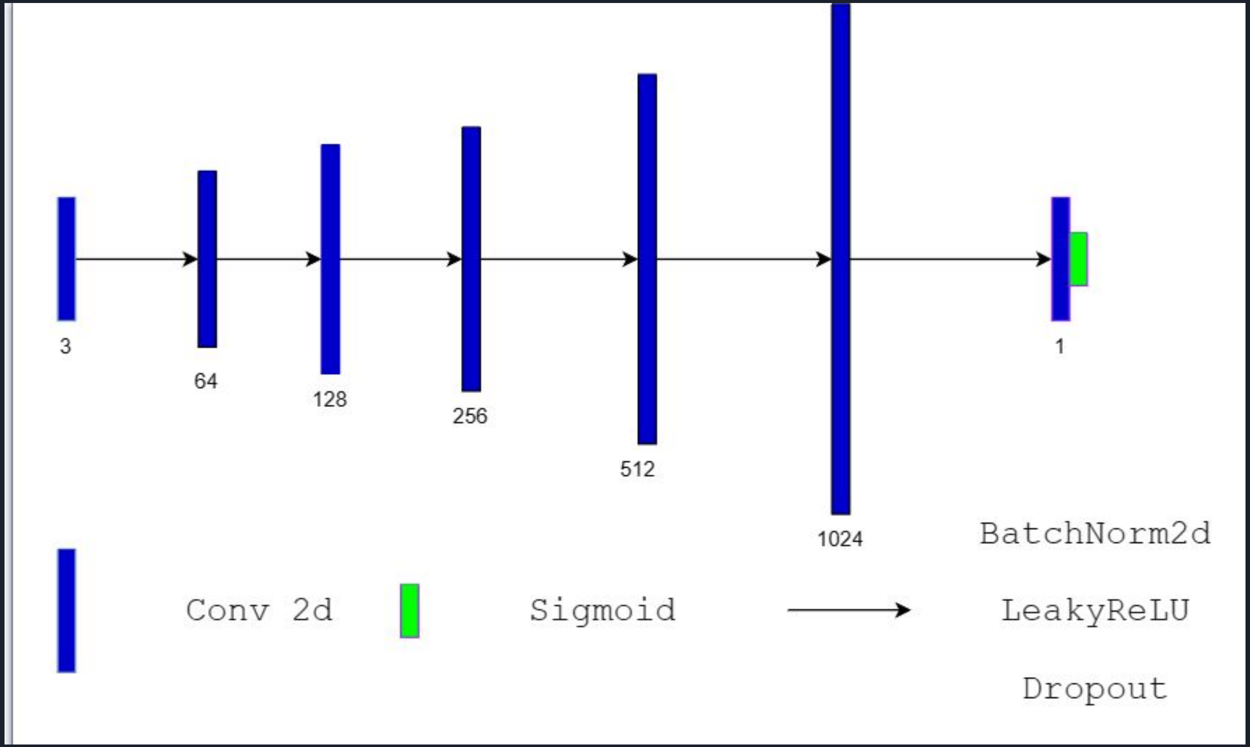
Generative Adversarial Model (GAN)

Generator U-net model

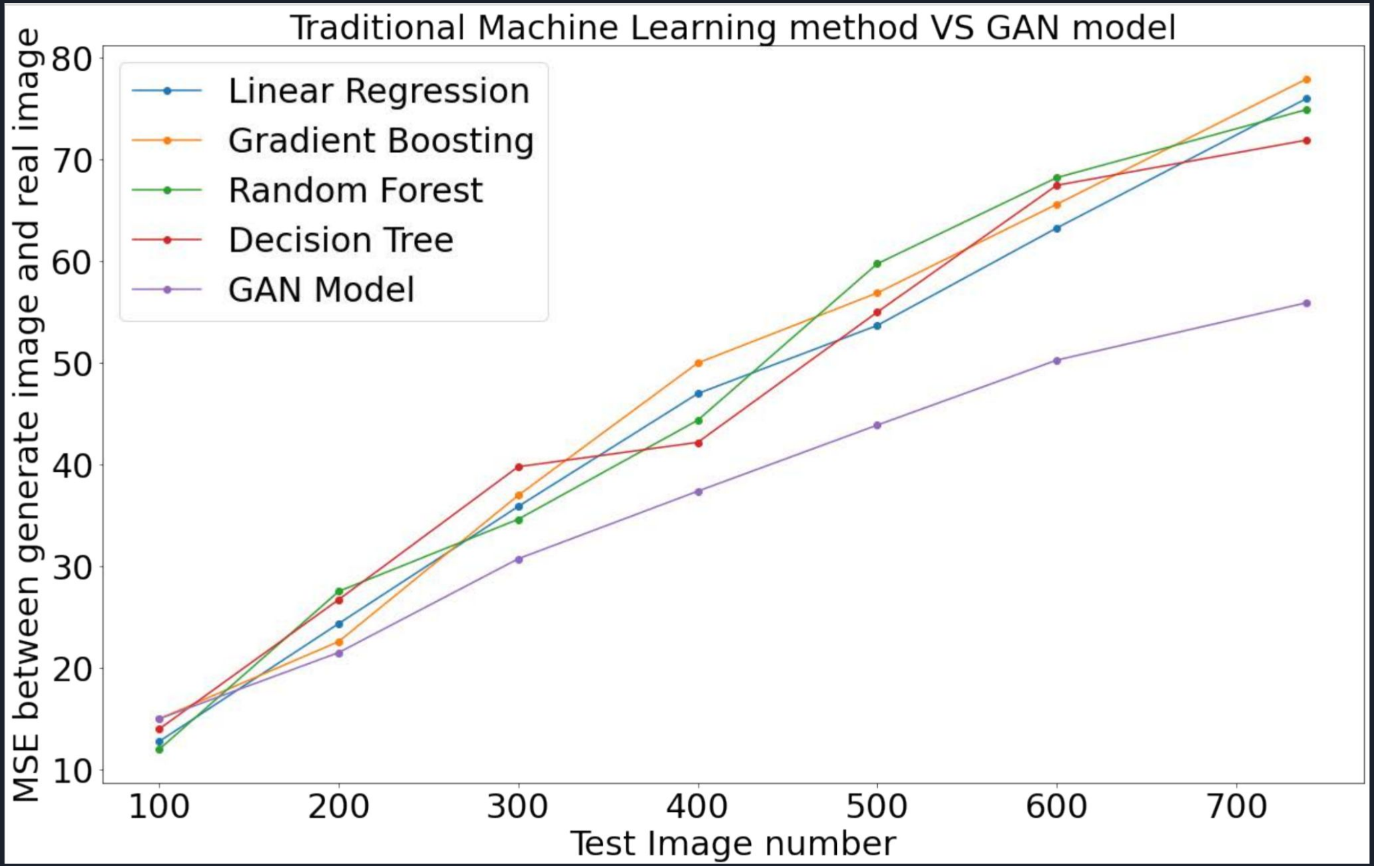


Generative Adversarial Model (GAN)

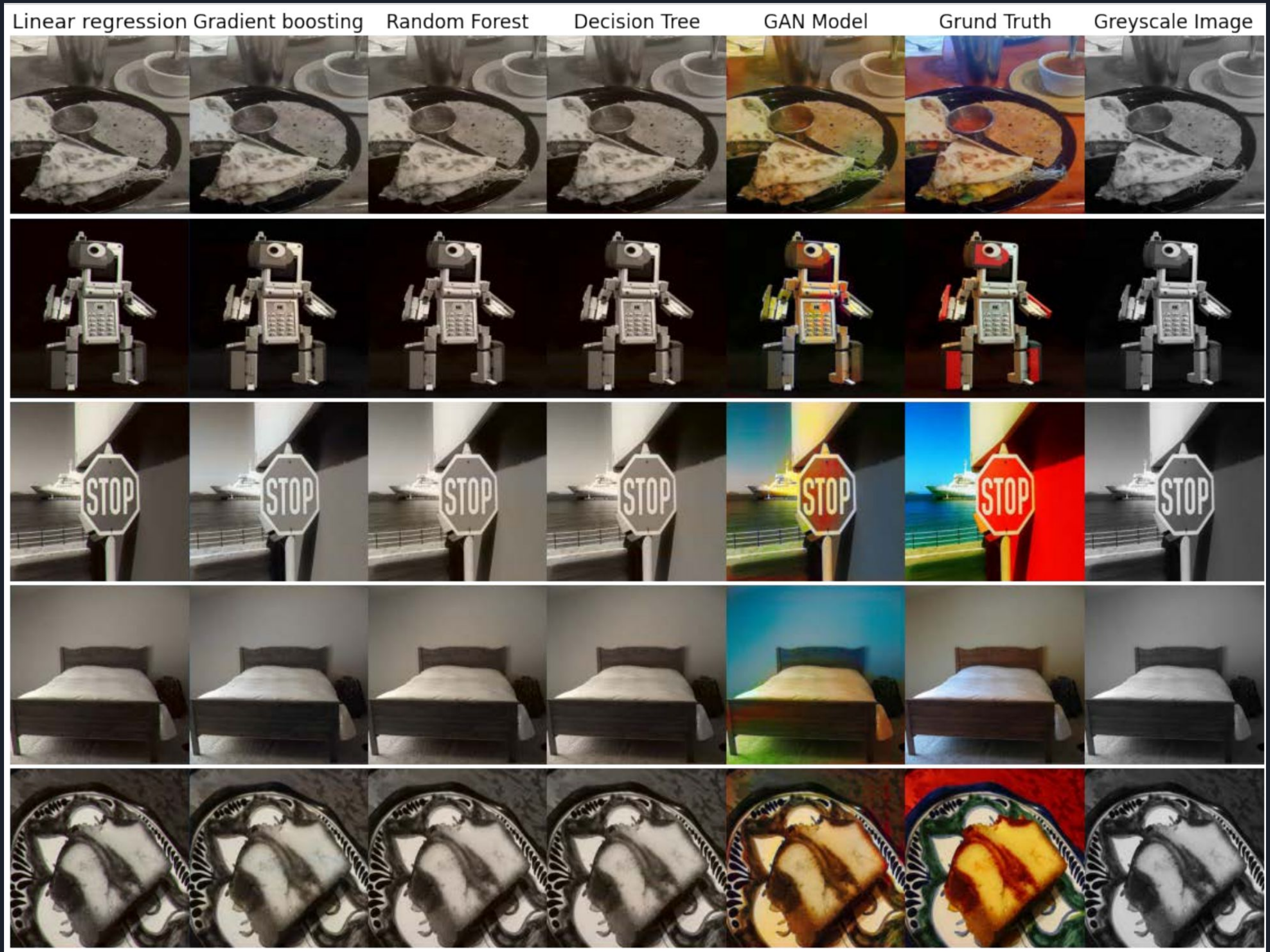
Discriminator model



Result


Traditional Machine Learning method VS GAN model
Linear Regression
Gradient Boosting
Random Forest
Decision Tree
GAN Model
MSE between generate image and real image
10
20
30
40
50
60
70
80
100
200
300
400
500
600
700
Test Image number

Result


Linear regression
Gradient boosting
Random Forest
Decision Tree
GAN Model
Grund Truth
Greyscale Image

Thank you!

Rainfall Prediction using Machine Learning

Based on data from Australia Bureau of Meteorology

Group 7 Bobby Bao, Kevin Li, Junrui Li

Motivation

Importance of rainfall prediction across sectors such as agriculture, urban planning, and emergency management, particularly in Australia's varied and often extreme weather conditions.

Introduction

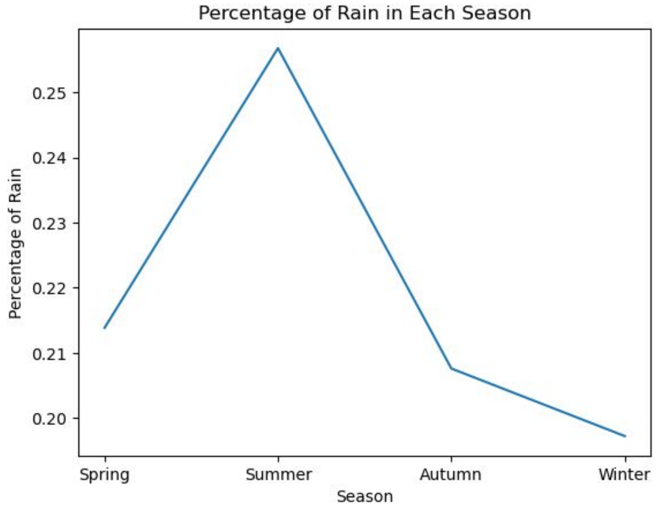
- Data Source: Australian Bureau of Meteorology dataset with a variety of meteorological parameters.
- Project Focus: Classify whether it will rain on the next day.
- Machine Learning Models Tested: Logistic Regression, Random Forests, Vanilla Neural Network, and XGBoost.
- Methodology: Compare results after dropping features with high collinearity
- Research Contribution: Determining the most effective machine learning model for binary rainfall prediction.

DATA DESCRIPTION

2008-2017 daily observation of 49 cities in Australia

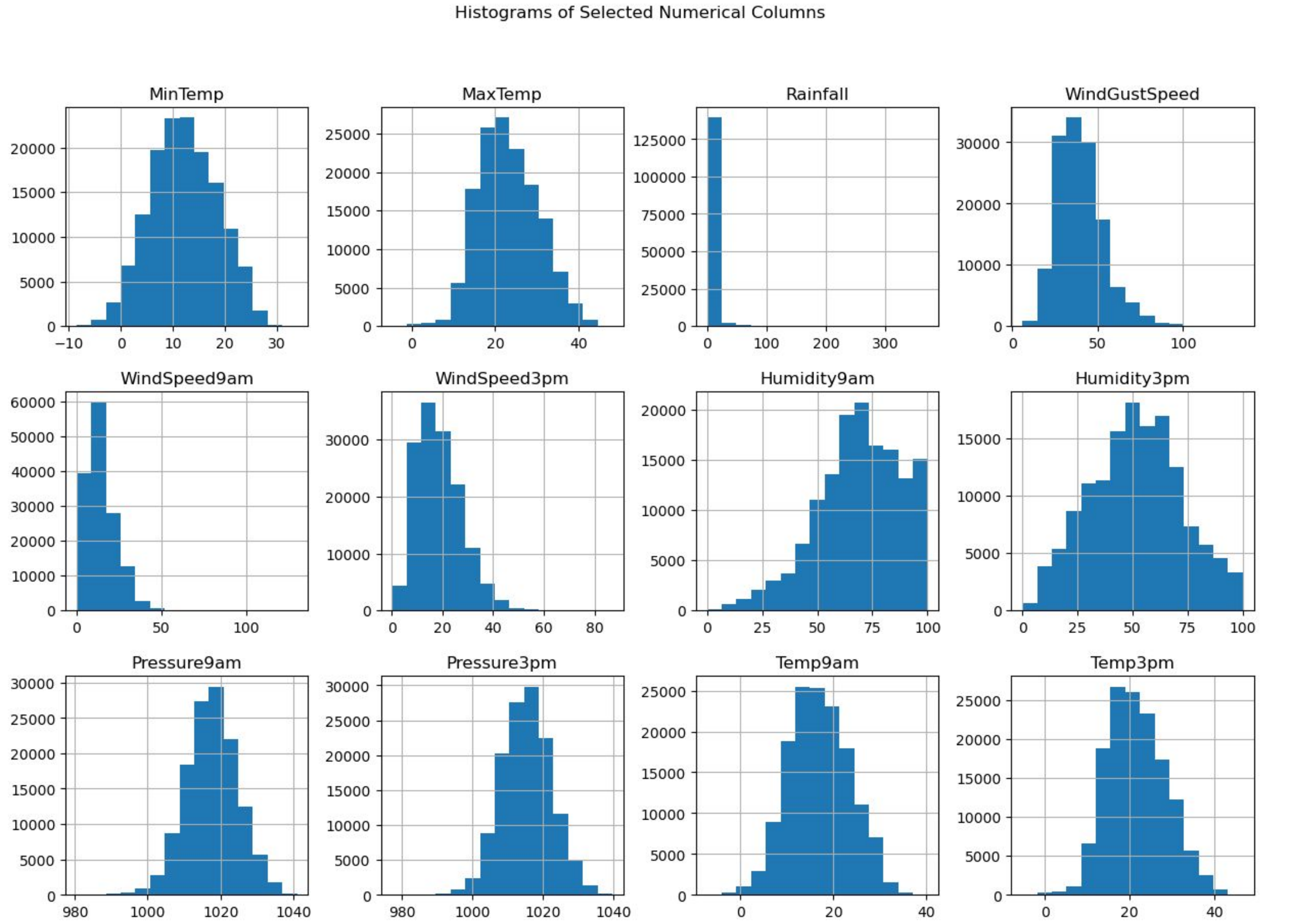
Original Data features

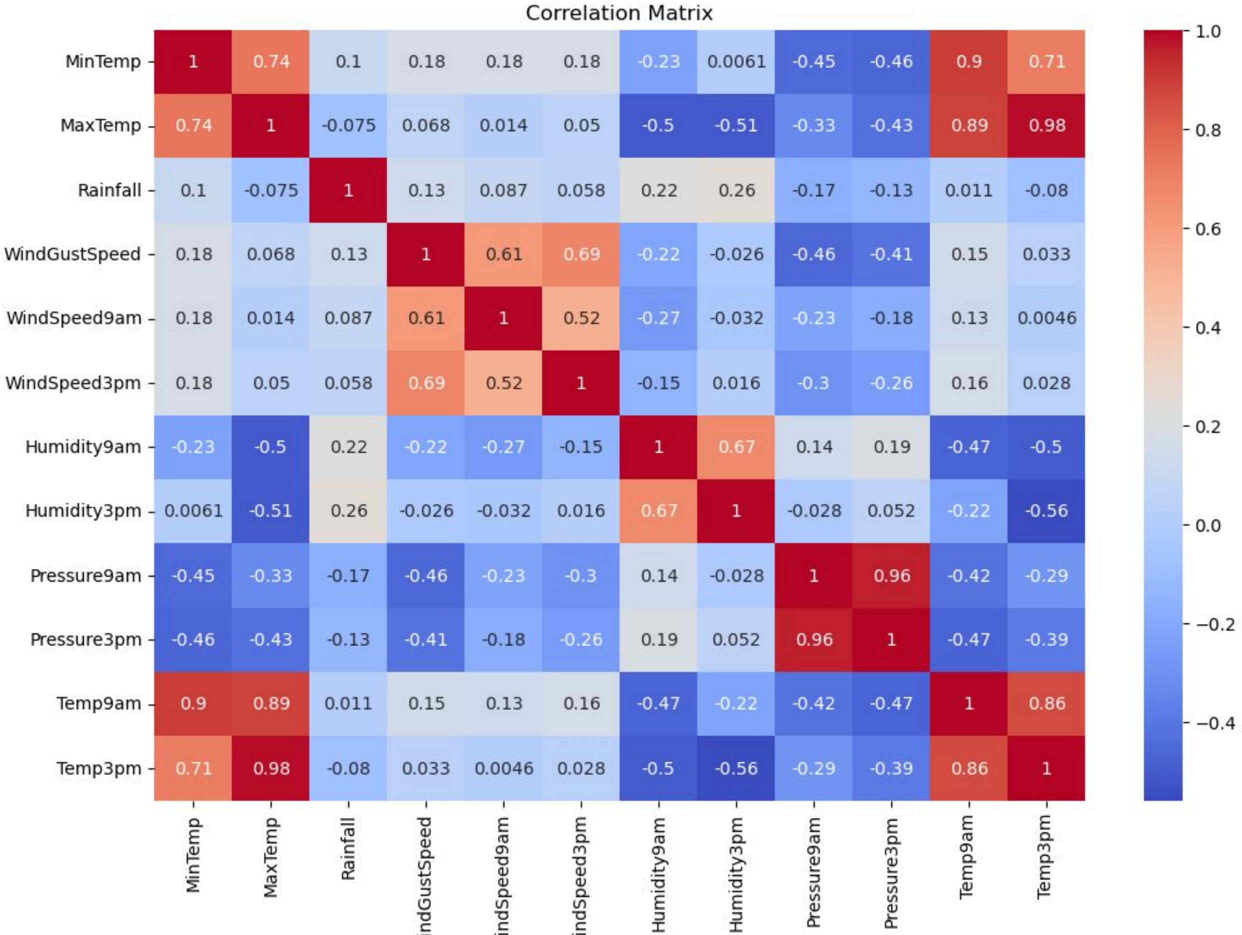
- *Date*
- *Location*
- *Temperature**
- *Rainfall*
- *Evaporation*
- *Sunshine*
- *Clouds**
- *Wind**
- *Humidity**
- *Pressure**



**At 9AM and 3PM / Min & Max*

Source:
https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package
http://www.bom.gov.au/climate/dwo/


Histograms of Selected Numerical Columns
MinTemp
MaxTemp
Rainfall
WindGustSpeed
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm
Pressure9am
Pressure3pm
Temp9am
Temp3pm


Correlation Matrix
MinTemp 1 0.74 0.1 0.18 0.18 0.18 -0.23 0.0061 -0.45 -0.46 0.9 0.71
MaxTemp 0.74 1 -0.075 0.068 0.014 0.05 -0.5 -0.51 -0.33 -0.43 0.89 0.98
Rainfall 0.1 -0.075 1 0.13 0.087 0.058 0.22 0.26 -0.17 -0.13 0.011 -0.08
WindGustSpeed 0.18 0.068 0.13 1 0.61 0.69 -0.22 -0.026 -0.46 -0.41 0.15 0.033
WindSpeed9am 0.18 0.014 0.087 0.61 1 0.52 -0.27 -0.032 -0.23 -0.18 0.13 0.0046
WindSpeed3pm 0.18 0.05 0.058 0.69 0.52 1 -0.15 0.016 -0.3 -0.26 0.16 0.028
Humidity9am -0.23 -0.5 0.22 -0.22 -0.27 -0.15 1 0.67 0.14 0.19 -0.47 -0.5
Humidity3pm 0.0061 -0.51 0.26 -0.026 -0.032 0.016 0.67 1 -0.028 0.052 -0.22 -0.56
Pressure9am -0.45 -0.33 -0.17 -0.46 -0.23 -0.3 0.14 -0.028 1 0.96 -0.42 -0.29
Pressure3pm -0.46 -0.43 -0.13 -0.41 -0.18 -0.26 0.19 0.052 0.96 1 -0.47 -0.39
Temp9am 0.9 0.89 0.011 0.15 0.13 0.16 -0.47 -0.22 -0.42 -0.47 1 0.86
Temp3pm 0.71 0.98 -0.08 0.033 0.0046 0.028 -0.5 -0.56 -0.29 -0.39 0.86 1
MinTemp
MaxTemp
Rainfall
indGustSpeed
indSpeed9am
indSpeed3pm
Humidity9am
Humidity3pm
Pressure9am
Pressure3pm
Temp9am
Temp3pm
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

Data preprocessing

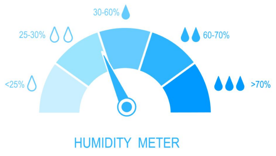
- Drop the non-numerical columns
- Drop features with missing data **over 30%**
- Use **Mean** to fill the missing data
- Feature normalization
- Group data by seasons
- Split **20%** of data as test set

Best Model Accuracy

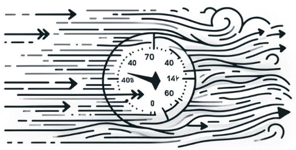
86% Average for 4 Seasons

XGBoost Feature Importance

Humidity at 3PM
32.3%

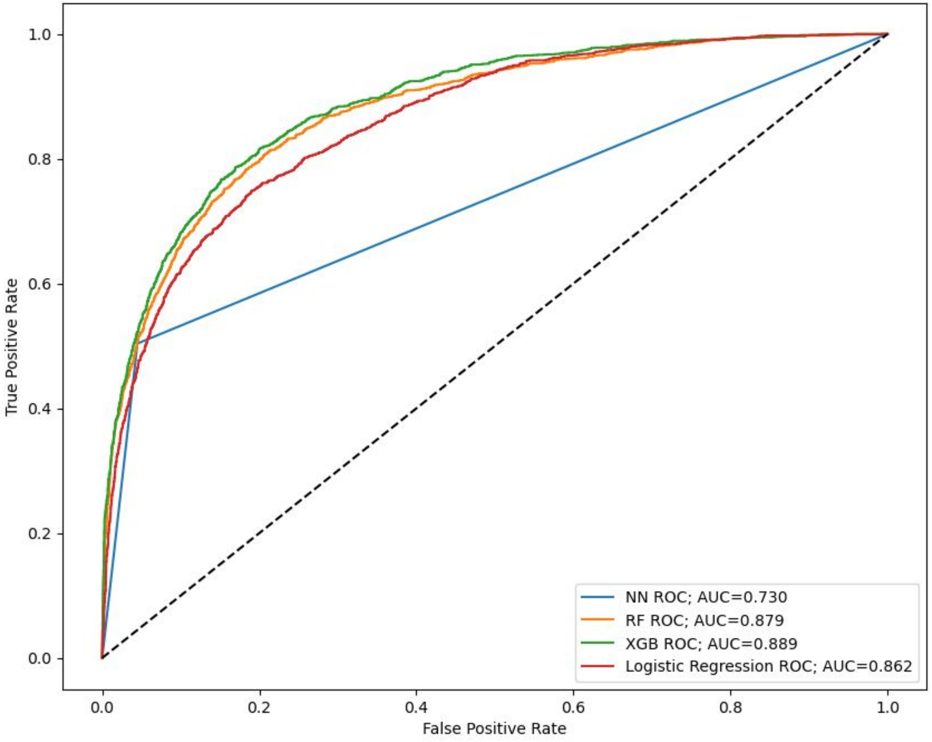


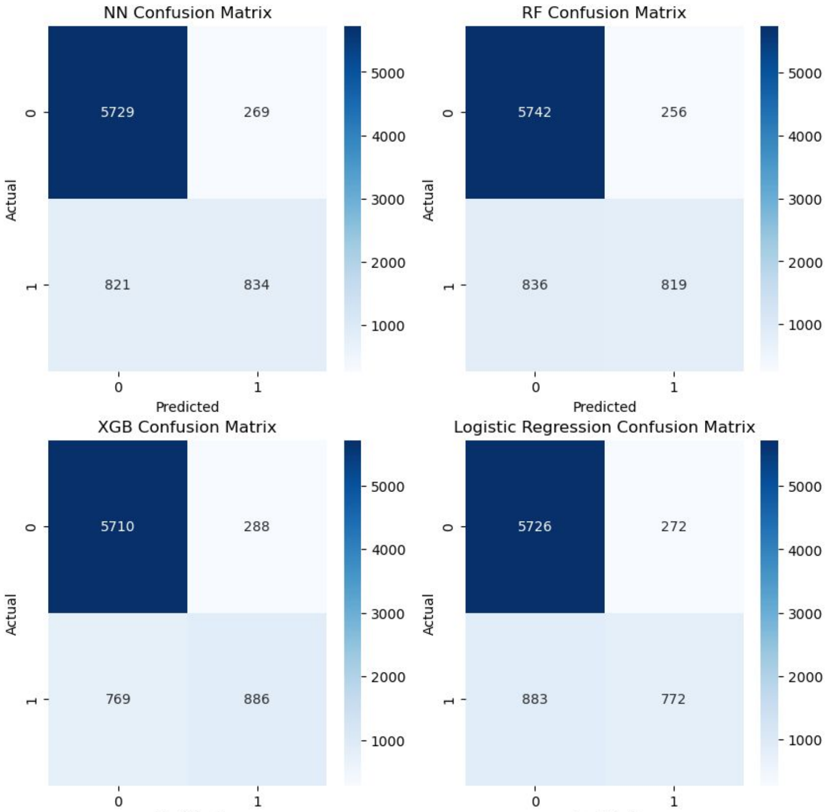
Wind Gust Speed
10.8%



Rainfall Today
7.9%

OUTPUT AND ACCURACY





Thank You!

NYC SHOOTING GEOGRAPHICAL ANALYSIS

Wen Yin

Group 8

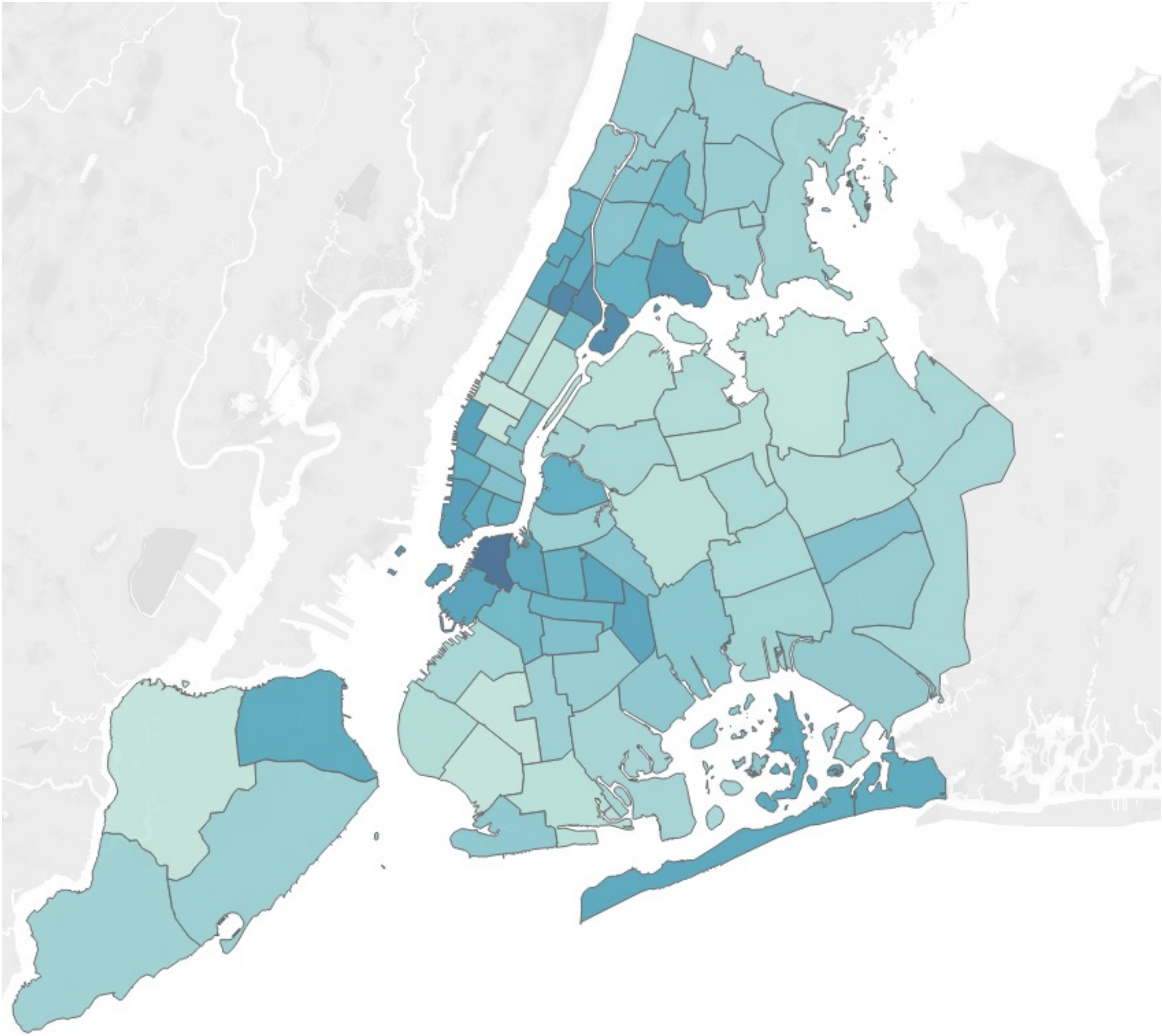
RESEARCH QUESTIONS

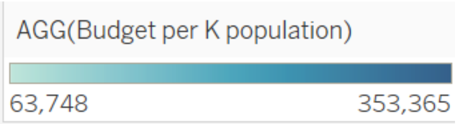
- We aim to examine the relationship between crime rates, socio-economic factors and police budget allocation at the precinct level.
- Does budget allocation vary based on crime rates and socio-economic factors. Which factors have the strongest impact on budget allocation.

INTRODUCTION

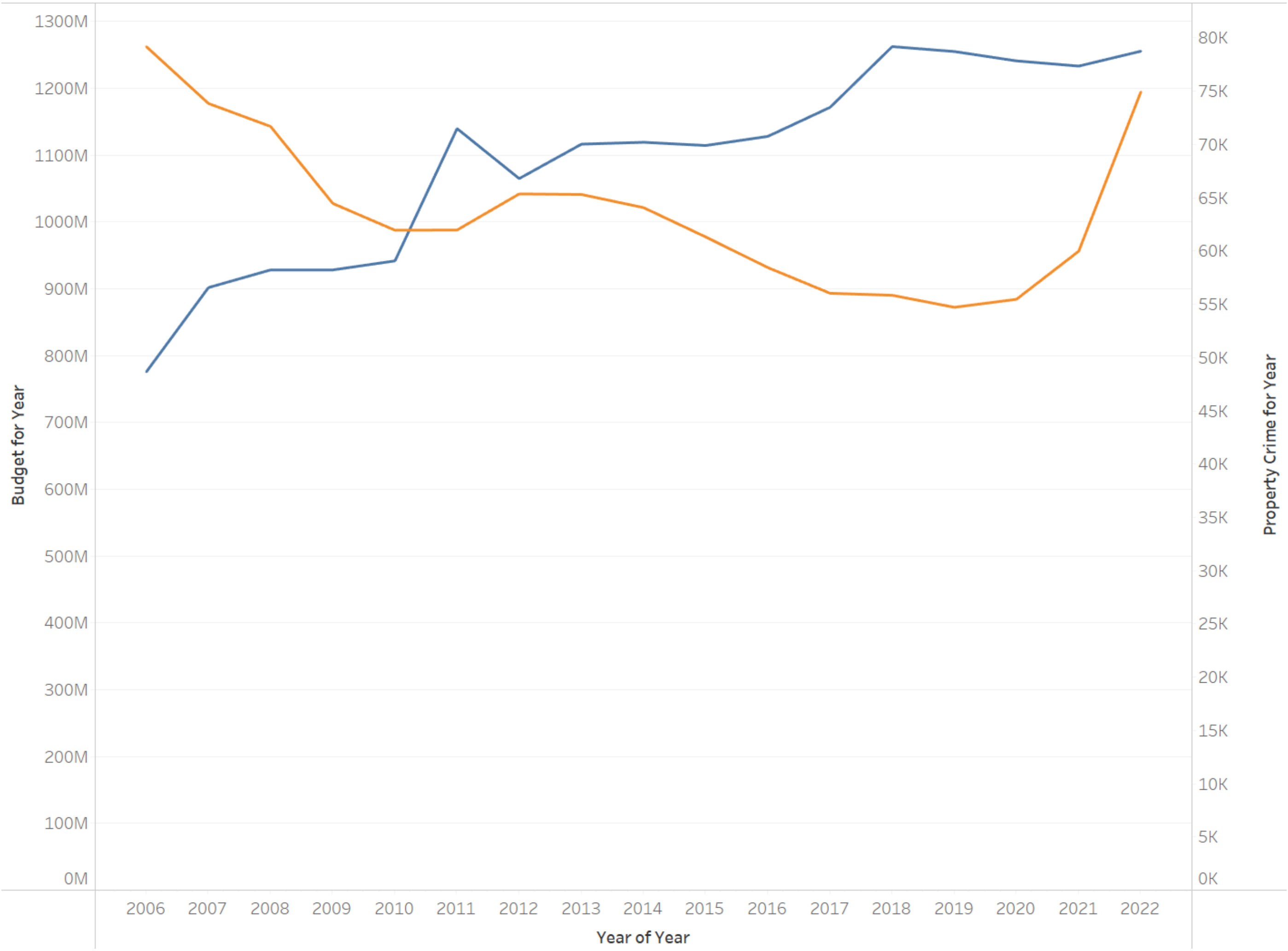
- This study aims to develop a data-driven model for estimating law enforcement budgets in urban areas, specifically focusing on New York City. It seeks to address the complex challenge of allocating funds to various precincts by analyzing crime rates, socio-economic factors, and historical data. Traditional methods of budget allocation, often based on intuition and historical patterns, do not adequately adapt to the evolving nature of urban crime. By examining the diverse neighborhoods and varying crime patterns in New York City, the study proposes a more nuanced approach to predict budget needs, shifting from conventional, intuitive decision-making to a robust, analytical methodology.

NEW YORK CITY POLICE BUDGET ALLOCATION ON PER CAPITA LEVEL



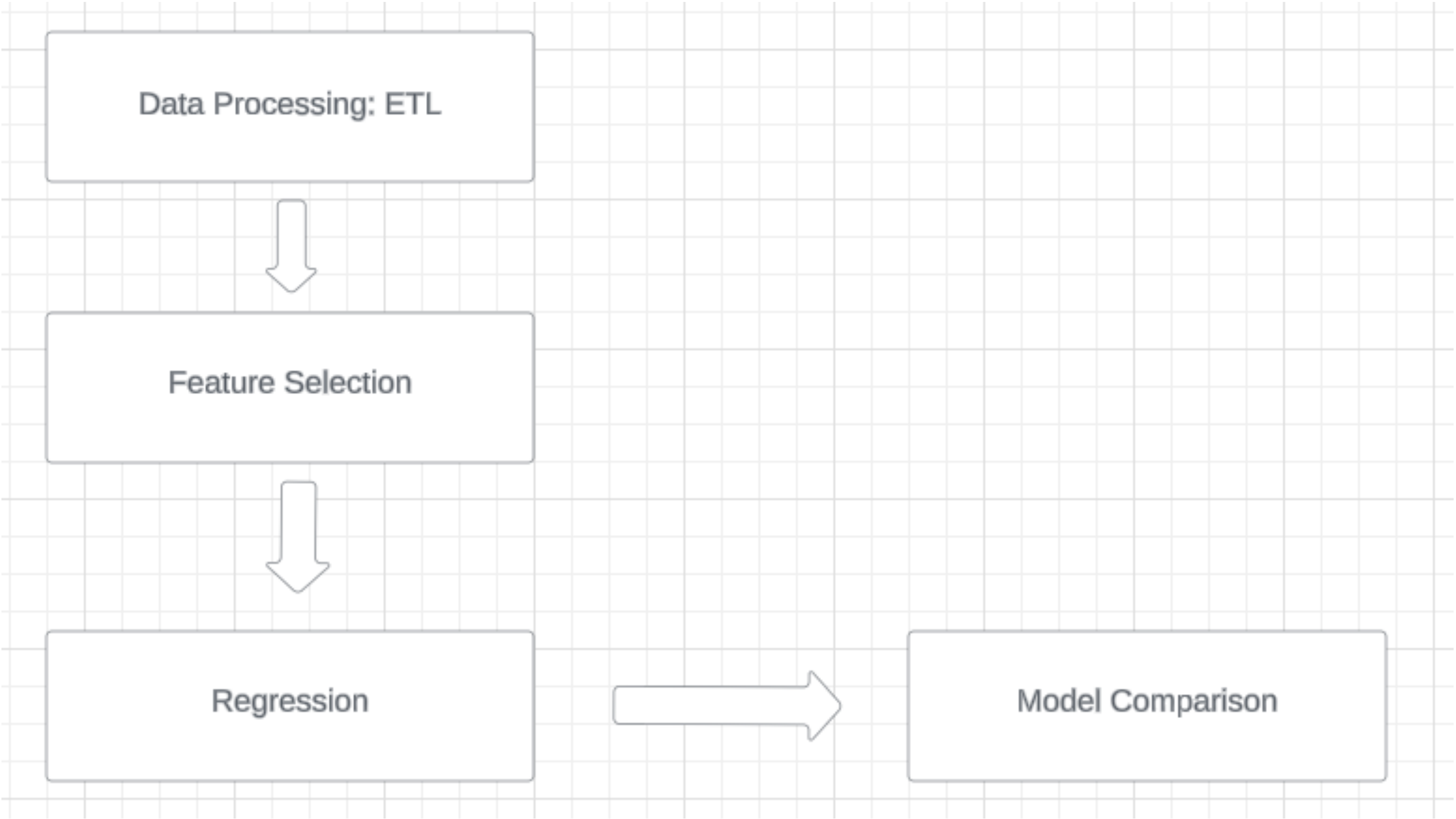


NYC POLICE BUDGET VS PROPERTY CRIME CASES



Measure Names
Budget for Year
Property Crime for Year

GENERAL FLOW CHART



DATA PROCESSING

Our study utilizes a comprehensive dataset amalgamating various sources to provide a holistic view of the factors influencing crime rates and budget allocation in New York City. The dataset comprises two primary components:

- Crime Dataset: Sourced from the New York Police Department (NYPD), this dataset spans from 2006 to 2021 and includes detailed statistics on crime types and percentages for each precinct, along with the NYPD's budget for the following year.
- Census Dataset: This dataset, derived from the American Community Survey for the years 2015 and 2017, provides socio-economic and demographic information at the census tract level. Key indicators include population demographics, income levels, employment status, and more.

Target: Adjusted_Budget_per_capit a

DATA PROCESSING

- Crime related: CRIME_Index_per_capita
- Census related: 'TotalPop','Men', 'Women', 'Citizen', 'Hispanic_', 'White_', 'Black_', 'Native_', 'Asian_’
- Income related: 'Employed', 'Poverty_', 'ChildPoverty_’, 'Adjust_IncomePerCap_’, 'Unemployment', 'Citizenship'
- Others: 'Professional’, 'Service', 'Office', 'Construction', 'Production', 'Drive', 'Carpool’, 'Transit', 'Walk', 'OtherTransp', 'WorkAtHome', 'MeanCommute’, 'PrivateWork', 'PublicWork', 'SelfEmployed', 'FamilyWork',

FEATURE SELECTION

Lasso

	Coefficient	Relationship
WorkAtHome	44.028352	Positive
CRIME_Index_per_capita	25.867925	Positive
PublicWork	14.804128	Positive
Asian_	11.476782	Positive
Construction	10.564184	Positive
Poverty_	8.571629	Positive
OtherTransp	7.766951	Positive
Men	6.295564	Positive
Full Time Positions	6.054733	Positive
Drive	3.341816	Positive
Year_2015	2.526530	Positive
precinct	0.273600	Positive
Walk	-0.000000	Negative
Year_2016	-0.000000	Negative
Black_	-0.000000	Negative
Citizenship	0.000000	Negative

Random Forest

	Coefficient	Relationship
Citizen	0.628277	Positive
TotalPop	0.093982	Positive
CRIME_Index_per_capita	0.072493	Positive
Employed	0.067793	Positive
Transit	0.033522	Positive
Full Time Positions	0.014592	Positive
Citizenship	0.012546	Positive
Asian_	0.010485	Positive
Black_	0.005438	Positive
Hispanic_	0.005292	Positive
Adjust_IncomePerCap_	0.004578	Positive
White_	0.003802	Positive
PrivateWork	0.003631	Positive
Production	0.003099	Positive
precinct	0.003052	Positive

FEATURE SELECTION

- Random Forest is the relative best feature selection method with least MSE

- Drop: 'Year_2015','Year_2016','Year_2017','Men','Unemployment','Professional','Women','Drive','MeanCommute'

REGRESSION AND MODEL COMPARISON

Table 1: MSE for different models

	Linear Regression	Decision Tree	SVM-RBF	SVM-Poly	SVM-Linear	Lasso	Neural Network
Without FS	701.43	859.23	538.29	767.49	625.03	699.29	323.24
With FS	808.95	711.86	390.96	746.88	863.00	785.46	**303.69**

REGRESSION AND MODEL COMPARISON

- In evaluating our machine learning models, we focused on their ability to predict law enforcement funding distribution in New York City utilizing social-economic indicators and crime rates. For this evaluation, the mean squared error (MSE) was employed as the major metric that measured how well the models worked.
- A detailed table shows huge disparities in the performances of the various models. Among them, the Neural Network model had the least MSE, even after FS. This shows its ability to address the complexity of associated relations when it comes to the modeling issues present in the data. However, other models such as linear regression and lasso regressions although may be very helpful in simple relationship scenarios were not efficient in this very complex situation presented.

FUTURE DIRECTION

- A focus in future research will be needed to improve these models by perhaps adding additional data sources or other more complicated machine learning approaches. Moreover, it is important to experiment with these models in non-urban settings just for the sake of testing their general applicability and adjustments to non-socio-economic urban conditions and crime patterns.

NYU

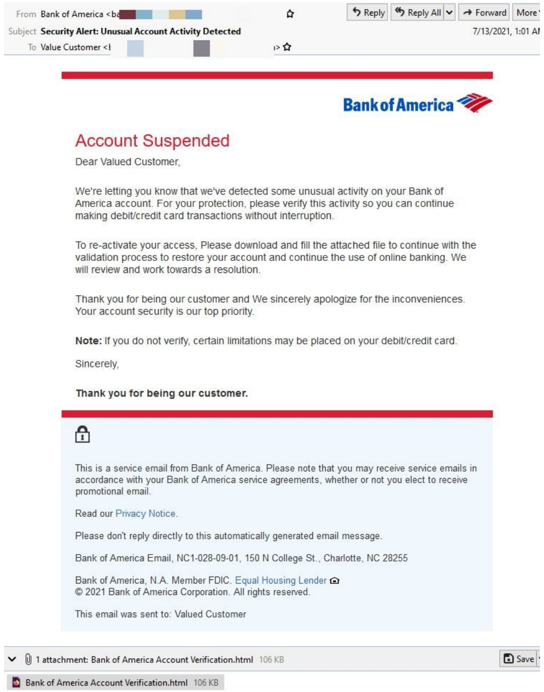
Optimizing Bank Account Fraud Detection

A comparative Study of ML Models and Ensemble Techniques (Group 10)

PRESENTED BY Xiangdong Zhang, Jiajun Jiao

12/12/2023

Topic Selection



From Bank of America <ba
Subject Security Alert: Unusual Account Activity Detected
To Value Customer <
Reply | Reply All | Forward | More
7/13/2021, 1:01 AM

Bank of America

Account Suspended

Dear Valued Customer,

We're letting you know that we've detected some unusual activity on your Bank of America account. For your protection, please verify this activity so you can continue making debit/credit card transactions without interruption.

To re-activate your access, Please download and fill the attached file to continue with the validation process to restore your account and continue the use of online banking. We will review and work towards a resolution.

Thank you for being our customer and We sincerely apologize for the inconveniences. Your account security is our top priority.

Note: If you do not verify, certain limitations may be placed on your debit/credit card.

Sincerely,

Thank you for being our customer.

This is a service email from Bank of America. Please note that you may receive service emails in accordance with your Bank of America service agreements, whether or not you elect to receive promotional email.

Read our Privacy Notice.

Please don't reply directly to this automatically generated email message.

Bank of America Email, NC1-028-09-01, 150 N College St., Charlotte, NC 28255

Bank of America, N.A. Member FDIC. Equal Housing Lender
© 2021 Bank of America Corporation. All rights reserved.

This email was sent to: Valued Customer

1 attachment: Bank of America Account Verification.html 106 KB
Save

Bank of America Account Verification.html 106 KB

FACTS:

- Google blocks around 100 million phishing emails daily.
- 71% of financial institutions reported a security breach from a business email compromise last year.

Fraud detection - CHALLENGING

- changing nature of fraud patterns over time
- limited availability of fraud examples to learn

NYU

Data Selection

Dataset used is composed of instances generated using a CTGAN, trained on a **real** bank account opening dataset, from Kaggle, protecting **privacy**.

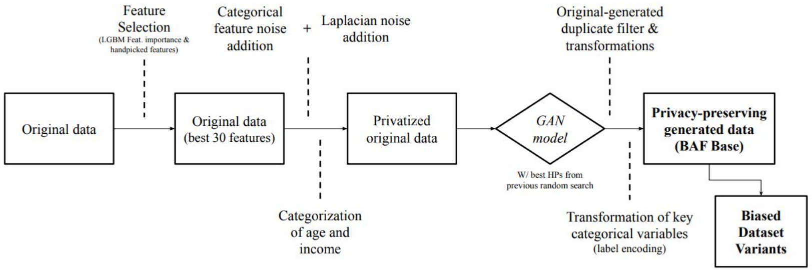


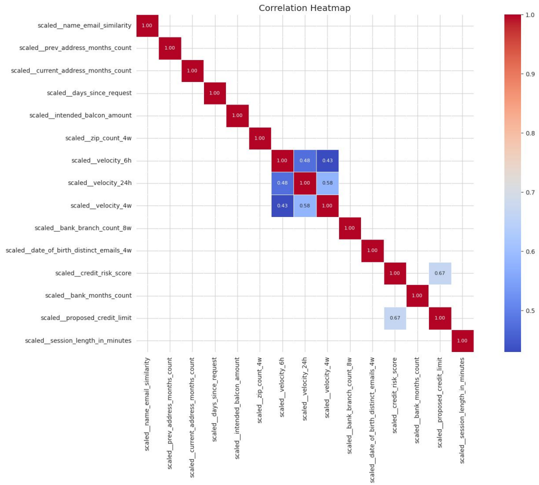
Figure 1: Illustration of the privacy-promoting interventions conducted.

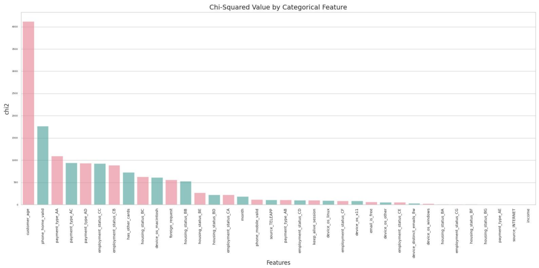
- **1 million** instances
- **30** pertinent features
- Diverse data
  - Mixed data types
  - ‘Month’ for Time-based pattern
  - **Personal info**: ‘age group’, ‘employment’, ‘income’
  - **Behavioral data**: ‘session length’, ‘transaction velocity’
- Class **imbalance**

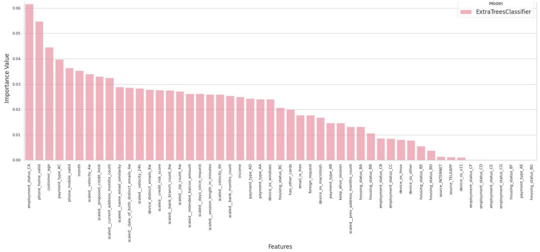
type	count
legitimate	988971
fraud	11029

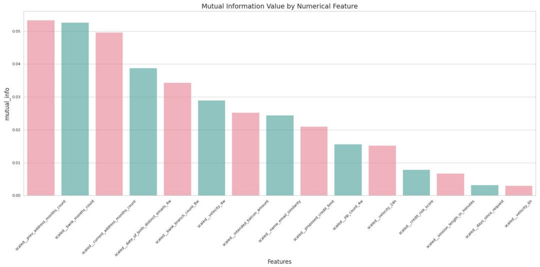
NYU

Exploratory Data Analysis









- Pearson correlation coefficient
- Chi-Squared Test for Categorical Features
- Mutual Information Test for Numeric Features
- Extra Trees Classifier for Feature Selection

NYU

Exploratory Data Analysis

SMOTE Oversampling

An algorithm used to augment the representation of the minority class in a dataset.

Grid Search CV

A method that uses stratified k-fold cross-validation to obtain the optimal hyperparameter for tuning of each classifier.

Exploratory Data Analysis

SMOTE is an oversampling algorithm proposed by JAIR in his 2002 article "SMOTE: Synthetic Over-Sampling Technique". In summary, this algorithm synthesizes new samples for a few classes based on interpolation.

If the number of samples for a minority class with a training set is T, the SMOTE algorithm will synthesize a new NT sample for that minority class. The requirement here is that N must be a positive integer, and if given N<1 then the algorithm will 'think' the number of samples of a few classes, T=NT, and forcedly set N=1, as in equation [6].

Consider a sample i for this minority class with a characteristic vector of x_i, $i \in \{1,...,T\}$.

- The k neighbours of sample x_i (e.g. Euclidean Distance) were first found in all t samples of this minority class, which are recorded as $x_{i(near)}$, $near \in \{1,...,k\}$.
- Then randomly select a sample $x_{i(nn)}$ from this k neighbour, and then generate a random number between 0 and 1, resulting in a new sample x_{i1}:

$$\boldsymbol{x}_{i1} = \boldsymbol{x}_i + \zeta_1 \cdot (\boldsymbol{x}_{i(nn)} - \boldsymbol{x}_i) \tag{1}$$

- Repeat Step 2 N times so that n new samples can be synthesized: x_{inew}, $new \in 1,...,N$

NYU

Models

Logistic Regression

$$f_{w,b}(x) = \frac{1}{1+e^{-(w\cdot x+b)}}$$

Decision Tree

$$H(p_1) = -p_1 \log_2(p_1) - (1-p_1)\log_2(1-p_1)$$

Random Forest

Extreme Gradient Boosting

$$L(t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i(t-1) + f_t(x_i)) + \Omega(f_t)$$

SVM

$$\min_{w,b,\xi} \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i\right)$$

LightGBM

$$L(\Theta) = \sum_{i=1}^{n} l(y_i, F(x_i;\Theta)) + \sum_{j=1}^{J} \Omega(f_j)$$

Neural Networks

$$L(y,\hat{y}) = -\sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

Evaluation

Logistic Regression

AUC-ROC **0.797**
Recall: **0.71**
Precision: **0.24**
F1 score: **0.36**

Decision Tree

AUC-ROC: **0.840**
Recall: **0.63**
Precision: **0.36**
F1 score: **0.46**

Random Forest

AUC-ROC: **0.922**
Recall: **0.68**
Precision: **0.52**
F1 score: **0.59**

Extreme Gradient Boosting

AUC-ROC: **0.949**
Recall: **0.68**
Precision: **0.74**
F1 score: **0.71**

SVM

AUC-ROC: **0.935**
Recall: **0.82**
Precision: **0.46**
F1 score: **0.59**

LightGBM

AUC-ROC: **0.952**
Recall: **0.68**
Precision: **0.77**
F1 score: **0.72**

Neural Networks

AUC-ROC: **0.936**
Recall: **0.34**
Precision: **0.88**
F1 score: **0.49**

* AUC-ROC: Area Under the Curve - ROC

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

NYU

Evaluation - Cont.

Decision Tree:
- Max Features: sqrt
- Max Depth: 10
- Criterion: entropy

Random Forest:
- Number of Estimators: 80
- Max Features: log2
- Max Depth: 10
- Criterion: entropy

XGB:
- Subsample: 0.8
- Number of Estimators: 100
- Min Child Weight: 4
- Max Depth: 8
- Learning Rate: 0.15
- ColSample ByTree: 1.0

LGB:
- Subsample: 0.8
- Number of Leaves: 50
- Number of Estimators: 500
- Max Depth: 7
- Learning Rate: 0.1
- ColSample ByTree: 0.8

Neural Networks:
- Sequential Model with:
  - Dense Layer: 256 units, ReLU activation, L2 Regularization (0.001)
  - Dropout: 0.6
  - Dense Layer: 128 units, ReLU activation, L2 Regularization (0.001)
  - Dropout: 0.6
  - Dense Layer: 1 unit, Sigmoid activation

NYU

Exploration

We weighted average **SVM** (highest fraud Recall: 0.82), **NN** (highest fraud Precision: 0.88) and **LightGBM** (highest AUC-ROC: 0.952 and fraud F1: 0.72):

1. **Weight**: {“nn”: 0.1, “svm”: 0.2, “lgbm”: 0.7}, **Threshold**: 0.1
   AUC-ROC: **0.953**, Precision: **0.88**, Recall: **0.34**, F1 score: **0.49**

2. **Weight**: {“nn”: 0.4, “svm”: 0.0, “lgbm”: 0.6}, **Threshold**: 0.4
   AUC-ROC: **0.823**, Precision: **0.82**, Recall: **0.66**, F1 score: **0.732**

NYU

Conclusion

- **Model Superiority**:
  Ensemble and boosted tree models have shown exceptional performance in fraud detection.
  SVM - highest fraud Recall, NN - highest fraud Precision: 0.88, LightGBM - highest AUC-ROC & F1

- **Integration Success**:
  Use of weighted average models enhanced detection capabilities.

- **Key Performance Metrics**:
  - High AUC-ROC scores confirm the models' ability to distinguish between classes.
  - Solid F1 scores highlight the models' proficiency in balancing precision and recall.
- **Key features that impact results**

Future Work

- **Task-Specific Model Configuration**:
  - Tailor models to prioritize either precision (minimizing false positives) or recall (minimizing missed fraud).
- **Research Opportunities**:
  - Explore strategies to balance precision, recall, and other performance metrics.
  - Investigate advanced data balancing techniques for improved model training.
- **Model Evolution**:
  - Continuous refinement of models to adapt to the changing landscape of fraud detection.
- **Impact on Fraud Detection**:
  - Enhancements aimed at bolstering security and trust in financial transactions.

Reference

European Central Bank. 6th report on card fraud, August 2020.

Nilson report. Card fraud losses reach $28.65 billion issue 1187, December 2020.

2006. [31] Lindsay CJ Mercer. Fraud detection via regression analysis. Computers & Security, 9(4):331–338, 1990.

Isangediok, Mary, and Kelum Gajamannage. "Fraud detection using optimized machine learning tools under imbalance classes." 2022 IEEE International Conference on Big Data (Big Data). IEEE, 2022.

Jesus, Sérgio, et al. "Turning the tables: Biased, imbalanced, dynamic tabular datasets for ml evaluation." Advances in Neural Information Processing Systems 35 (2022): 33563-33575.

Nader Mahmoudi and Ekrem Duman. Detecting credit card fraud by modified fisher discriminant analysis. Expert Systems with Applications, 42(5):2510–2516, 2015.

Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit, pages 60–69, 2007.

Arun Prakash and Chandramouli Chandrasekar. An optimized multiple semi-hidden markov model for credit card fraud detection. Indian Journal of Science and Technology, 8(2):176, 2015.

Sangeeta Mittal and Shivani Tyagi. Performance evaluation of machine learning algorithms for credit card fraud detection. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pages 320–324. IEEE, 2019.

Victoria Priscilla and Padma Prabha. Credit card fraud detection: A systematic review. In International Conference on Information, Communication and Computing Technology, pages 290–303. Springer, 2019.

Imane Sadgali, Nawal Sael, and Faouzia Benabbou. Detection of credit card fraud: State of art. Int. J. Comput. Sci. Netw. Secur, 18(11):76–83, 2018.

Janvier Omar Sinayobye, Fred Kiwanuka, and Swaib Kaawaase Kyanda. A state-of-the-art review of machine learning techniques for fraud detection research. In 2018 IEEE/ACM symposium on software engineering in africa (SEiA), pages 11–19. IEEE, 2018

Yann-Ael Le Borgne, Wissam Siblini, Bertrand Lebichot, and Gianluca ¨ Bontempi. Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook. Universite Libre de Bruxelles, 2022.

Charles X Ling and Victor S Sheng. Cost-sensitive learning and the class imbalance problem. Encyclopedia of machine learning, 2011:231–235, 2008

Sam Maes, Karl Tuyls, Bram Vanschoenwinkel, and Bernard Manderick. Credit card fraud detection using bayesian and neural networks. In Proceedings of the 1st international naiso congress on neuro fuzzy technologies, volume 261, page 270, 2002.

Thank you!

NYU

NYU

Investigating the Effect of Data Quality on Breast Cancer Prediction

Animesh Ramesh (ar8006)
Caraline Bruzinski (cb4904)
Tony Kimathi (tkk8363)

Problem Statement

Our project focused on developing a predictive system for breast cancer using machine learning.
We faced the challenge of analyzing two versions of the Wisconsin Breast Cancer dataset: the original 1992 uncleaned data and the 1995 cleaned data.
There were several analyses of the 1995 data but few analyzed the 1992 data with missing values. **We were curious to see the effect of having missing values on breast cancer prediction.**
The goal was to compare the effectiveness of logistic regression, SVM, and bayesian models, highlighting the impact of data quality on the accuracy of health predictions.

NYU

Datasets

1992 Dataset: This dataset is an earlier version of the Wisconsin Breast Cancer dataset. It includes features such as clump thickness, uniformity of cell size and shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitosis. These features are primarily numerical, extracted from digitized images of breast mass cell samples. Each entry also includes a diagnosis, indicating whether the observed cell mass is benign or malignant. This dataset is notable for being in a less processed or 'uncleaned' state, presenting initial challenges in data quality and completeness.

(9 features, 699 rows)

1995 Dataset: This is a more refined version of the Wisconsin Breast Cancer dataset. It contains a more comprehensive set of features including radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension, measured in three contexts: mean, standard error, and worst or largest (mean of the three largest values). Like the 1992 dataset, it includes a diagnosis label. The 1995 dataset represents a more processed or 'cleaned' state, providing a more detailed and refined set of features for analysis.

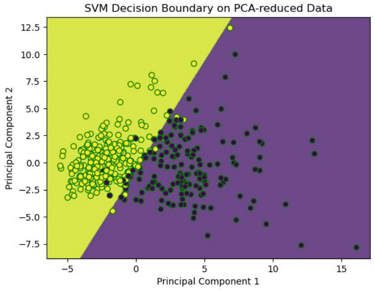
(30 features, 569 rows)

Cleaning the Dataset

- In processing the 1992 breast cancer dataset, a structured approach was employed to address missing values. Each column was scrutinized for NaNs.
- Columns with no missing values were retained as is. For those with missing data, a tailored strategy based on distribution characteristics was applied.
- The skewness of each column was evaluated; columns exhibiting a skewness below 0.5 and having fewer than 10% zero values were replenished using the **mean**, preserving the central tendency of the data.
- Conversely, in columns where zeros constituted over 10% of the data, a **zero-filling** approach was adopted, recognizing the prevalence of zero values.
- For columns with more pronounced skewness, **the median** was utilized, apt for handling skewed data distributions effectively.
- The 'Bare Nuclei' column, uniquely identified for its missing values in this dataset, received focused attention in this imputation process, ensuring the dataset's readiness for subsequent machine learning applications.

THE SVM MODEL

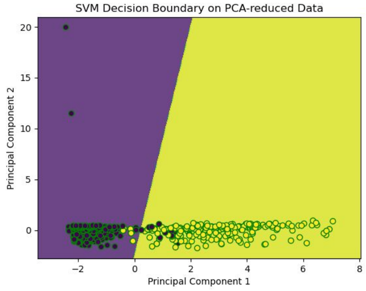
- Accuracy:
  - 1992 dataset: 97.14%.
  - 1995 dataset: 98.25%.
- Precision:
  - 1992: 97% for benign, 98% for malignant.
  - 1995: Similarly high.
- Recall:
  - 1992: 99% for benign, 93% for malignant.
  - 1995: 100% for malignant, 95% for benign.
- F1-Scores: Consistently high across both datasets, indicating balanced accuracy.
- Hyper Parameter Grid:
  - Values for 'C': [0.1, 1, 10, 100, 1000].
  - Optimal 'C' Value: 0.1
- Data Quality and Features:
  - Superior performance in the 1995 dataset suggests higher data quality and more informative features compared to 1992.



1995 Dataset

Metric	Class 0	Class 1	Overall (Weighted Avg.)
Precision	100%	97%	98%
Recall	95%	100%	98%
F1-Score	98%	99%	98%
Support	43	71	114

- **Overall Accuracy:** 98.25%
- **Confusion Matrix:**
  - True Positives for Class 0: 41
  - False Positives: 2
  - False Negatives: 0
  - True Negatives for Class 1: 71



1992 Dataset

Metric	Benign Cases (Class 2)	Malignant Cases (Class 4)	Overall (Weighted Avg.)
Precision	97%	98%	97%
Recall	99%	93%	96%
F1-Score	98%	95%	97%
Support	95	45	140

- **Overall Accuracy:** 97.14%
- **Confusion Matrix:**
  - True Positives for Benign: 94
  - False Positives: 1
  - False Negatives: 3
  - True Negatives for Malignant: 42

NYU

The Bayesian Regression Model

$$p(\theta|D) \propto p(\theta) \cdot p(D|\theta)$$

$$b_0 \sim N(\mu = 0, \sigma = 10)$$

$$\theta \sim N(\mu = 0, \sigma = 10)$$

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
b0	-2.90	2.15	-6.64	1.11	0.10	0.07	495.0	423.0	1.01
thetas[0]	1.21	8.10	-11.58	19.01	0.25	0.28	1035.0	613.0	1.00
thetas[1]	-1.31	2.14	-5.32	2.60	0.09	0.07	506.0	416.0	1.01
thetas[2]	2.60	7.92	-11.53	18.30	0.25	0.31	976.0	559.0	1.00
thetas[3]	-0.26	8.11	-16.28	14.16	0.25	0.24	1032.0	685.0	1.00
thetas[4]	-0.95	2.67	-6.00	4.11	0.11	0.08	556.0	510.0	1.01
thetas[5]	15.67	5.48	5.37	25.94	0.22	0.17	652.0	403.0	1.01
thetas[6]	-7.00	5.83	-17.41	4.64	0.22	0.17	744.0	526.0	1.01

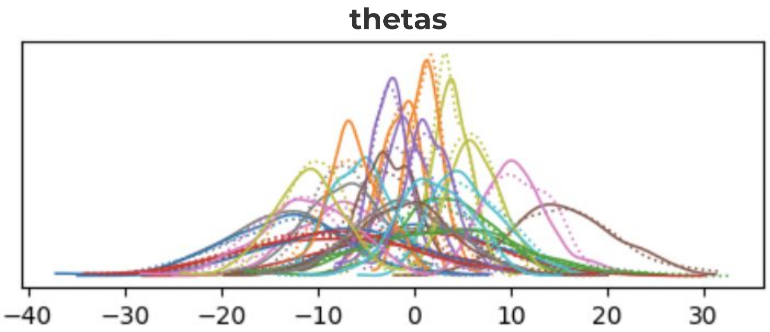
Bayesian modeling captures uncertainty in the data; it integrates over the entire posterior distribution. Inference is made in terms of probabilities

Priors: Non-informative, **gaussian priors**

Likelihood: Binary outcome, follows a **bernoulli distribution**

The posterior is the updated belief about theta, given data we have observed

Draw samples from the posterior distribution: MCMC (Markov Chain Monte Carlo) sampling, vs MAP estimate



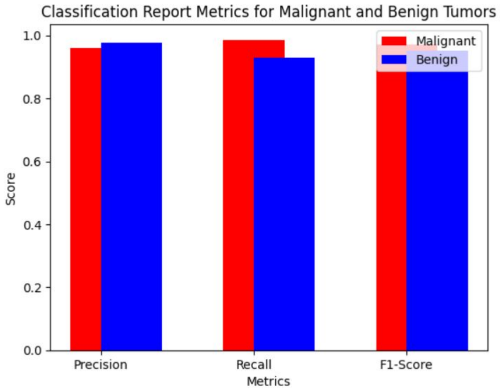
(1992 - missing values)
Accuracy 96.42%
ROC-auc 0.996

(1995 - enriched)
Accuracy 95.61%
ROC-auc 0.993

Logistic Regression Model

- Used 20% for test size & 42 random state
- The odds ratio is estimated by taking the exponential of the model coefficients (eg, exp[β1]).

1995

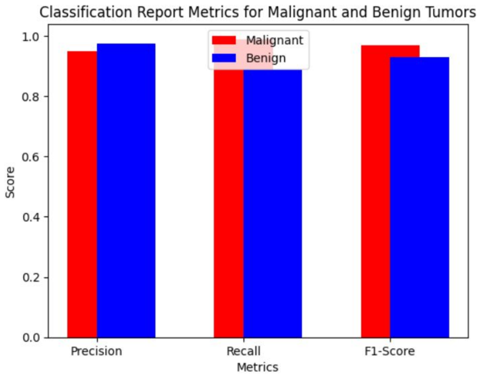


Overall: 96.49% accuracy

- TN: 40
- FP: 3
- FN: 1
- TP: 70

Metric	Class 0 (Benign)	Class 1 (Malignant)	Overall
Precision	97.56%	95.89%	96.49%
Recall	93.02%	98.59%	95.81%
F1-Score	95.24%	97.22%	96.23%
Support	43	71	114

1992



Overall: 95.71% accuracy

- TN: 94
- FP: 1
- FN: 5
- TP: 40

Metric	Class 0 (Benign)	Class 1 (Malignant)	Overall
Precision	94.95%	97.56%	95.71%
Recall	98.95%	88.89%	93.92%
F1-Score	96.91%	93.02%	94.97%
Support	95	45	140

Comparison Results

Overall Experience

- SVM was the **best performing model** among the three models (98%).
- Bayesian model performed **better on 1992 dataset** with missing values, than on the enriched 1995 dataset
- Having missing values in the data **did not *significantly*** affect the breast cancer prediction.

	1995 dataset (*enriched*)	**1992 dataset** (*original*)
Bayesian	95.61	96.42
SVM	98.25	97.14
Logistic	96.49	95.71

Related work [6,36] results (1995 data):

Algorithms	Accuracy (%)
Naive Bayes	93.75
Support Vector Machine	96.25
Logistic Regression	95.0

NYU

References

1. Performance Comparison of Different Machine Learning Techniques for Early Prediction of Breast Cancer Using Wisconsin Breast Cancer Dataset
2. https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
3. https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original
4. Bayesian Reasoning and Machine Learning textbook here
5. https://statswithr.github.io/book/introduction-to-bayesian-regression.html
6. https://www.researchgate.net/publication/343436577_Selecting_Features_for_Breast_Cancer_Analysis_and_Prediction

NYU

Thank you!

Questions?

Github:
https://github.com/Animesh-Ramesh/BreastCancerPrediction

MOVIE RATING PREDICTOR

Anuva Sehgal
Ravan Buddha

CONTENTS

INTRODUCTION

- Forecasting movie ratings through various regression techniques using a comprehensive dataset.
- Employing a multitude of data preprocessing techniques to prepare the data.
- Unveiling insights crucial for understanding audience preferences and film success metrics.
- Analyzing how movie features influence audience voting patterns: Positively or Negatively.

DATA

"MOVIES"

This dataset contains essential movie details like budget, genres, popularity, etc.

“CREDITS”

This dataset includes movie IDs, titles, cast, and crew information.

DATA INTEGRATION

- Both the datasets are merged together based on Movie ID.
- The dataset only contains English-Language movies for comprehensive analysis.
- This decision also has minimal impact as there are very few non-English movies in the dataset.

DATA PREPROCESSING

Preparing the data

PREPROCESSING

Feature Selection	JSON to String	Genre Representation	Popularity	Text Data Integration
Unnecessary features like homepage, status etc. are removed.	Features like Overview, genres are in JSON format, converted them into strings.	Each unique genre will be a feature and contains the value 1 if that movie belongs to that genre.	Popularity of actors, director and production companies are calculated.	Features Overview, Keywords, and Tagline are merged into a unified 'tags' column.

POPULARITY

- Popularity is the sum of the (average_vote * vote_count) across all the movies they have appeared in.
- Actor Popularity – The popularity of the first 3 actors from the cast.
- Director Popularity
- Production Companies Popularity

‘TAGS’ FILTERING

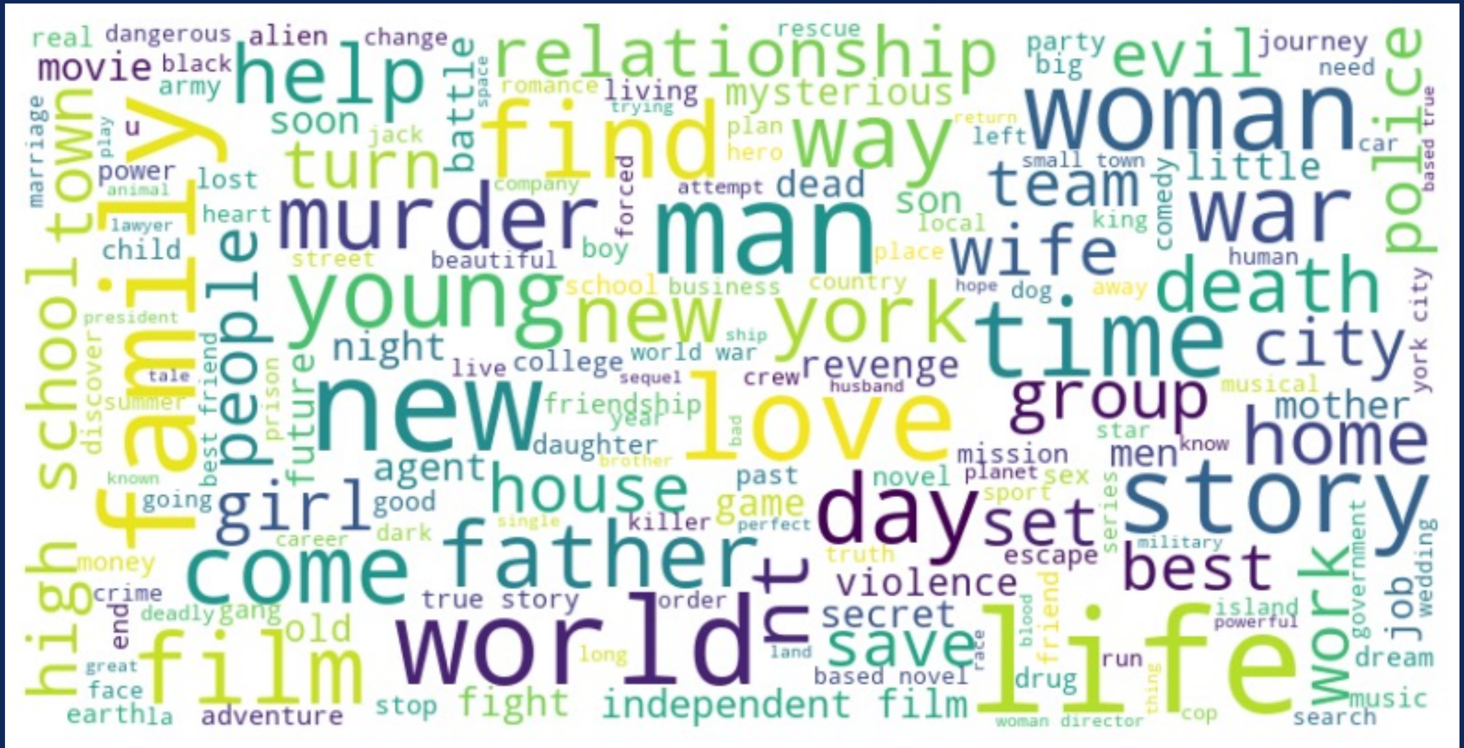
NLTK

- Stopwords Filter
- WordNetLemmatizer
- Custom Filters

SpaCy

- Removal of Unimportant words
- en_core_web_sm dictionary

WORD CLOUD OF TAGS AFTER FILTERING



‘TAGS’ ANALYSIS

TF-IDF (TERM FREQUENCY – INVERSE DOCUMENT FREQUENCY)

- Purpose

  Measures the importance of a term in a document relative to a collection of documents. Represents the significance of a word by considering how often it appears in a document (term frequency) and how uncommon it is across all documents (inverse document frequency).

- Usage

  Converted textual data into numerical vectors, emphasizing important words while downplaying common words.

TRUNCATED SVD(SINGULAR VALUE DECOMPOSITION)

- Purpose

  Reduces the dimensionality of a matrix by finding a lower-dimensional representation that captures the most important patterns or relationships in the data. Matrix Factorization.

- Usage

  Applied in various fields, including NLP, to transform high-dimensional data into a lower-dimensional space, often used after vectorization techniques like TF-IDF to further compress and capture essential information.

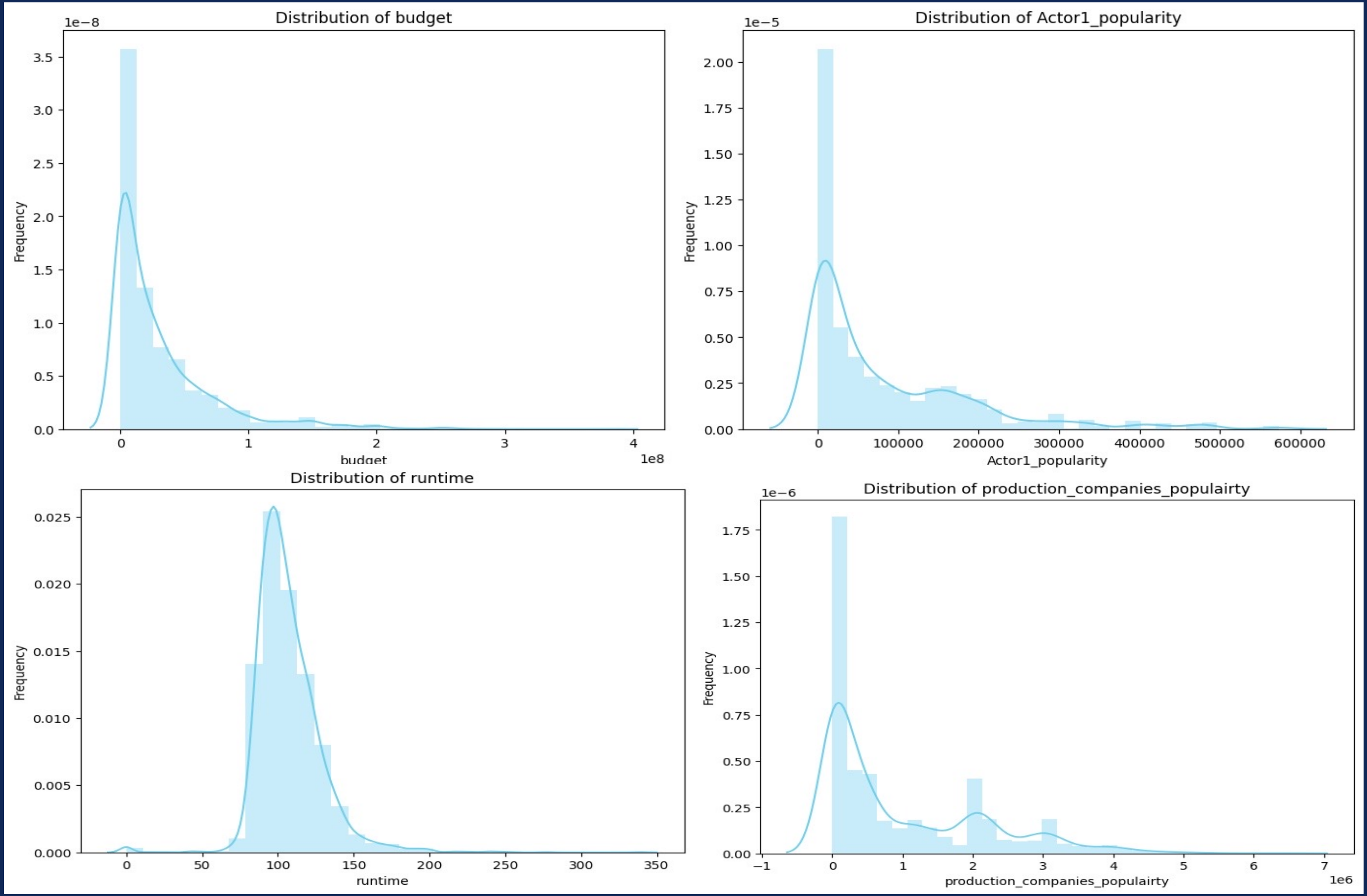
REASON FOR USING TF-IDF AND TRUNCATED SVD

- Combining TF-IDF with Truncated SVD helps manage high-dimensional text data efficiently. TF-IDF initially captures word importance, while Truncated SVD reduces this representation's dimensionality without losing significant information, improving computational efficiency.
- The joint application of TF-IDF and Truncated SVD can enhance the performance of downstream machine learning models by providing a more compact yet informative representation of the text data.

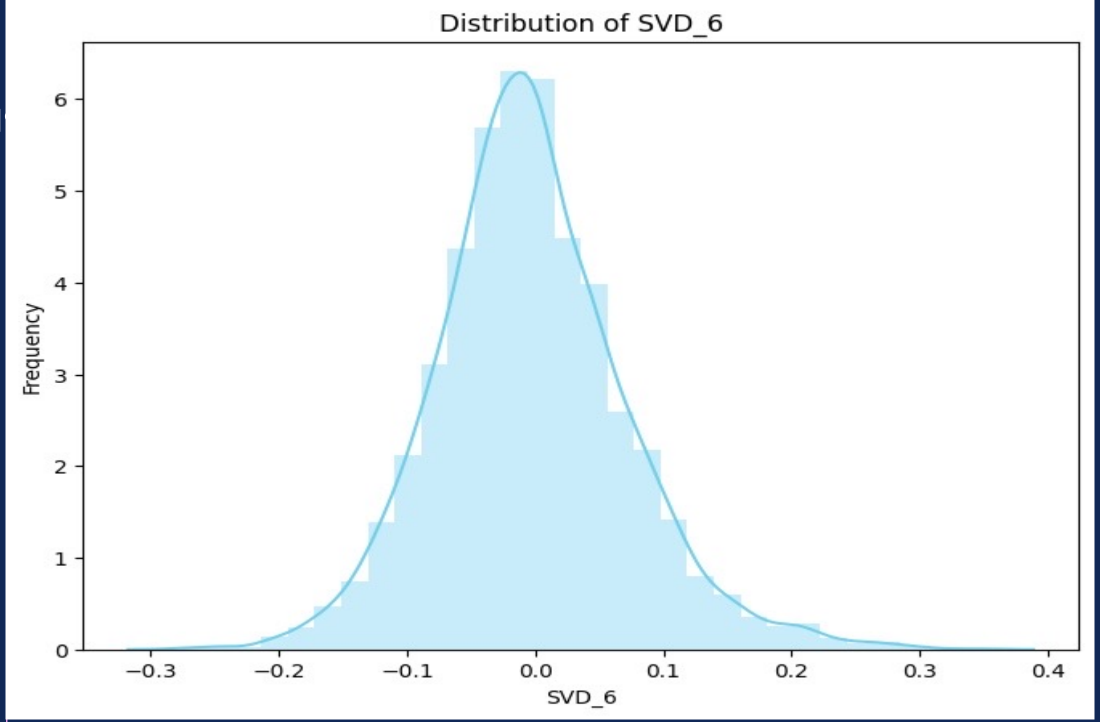
FEATURE SCALING

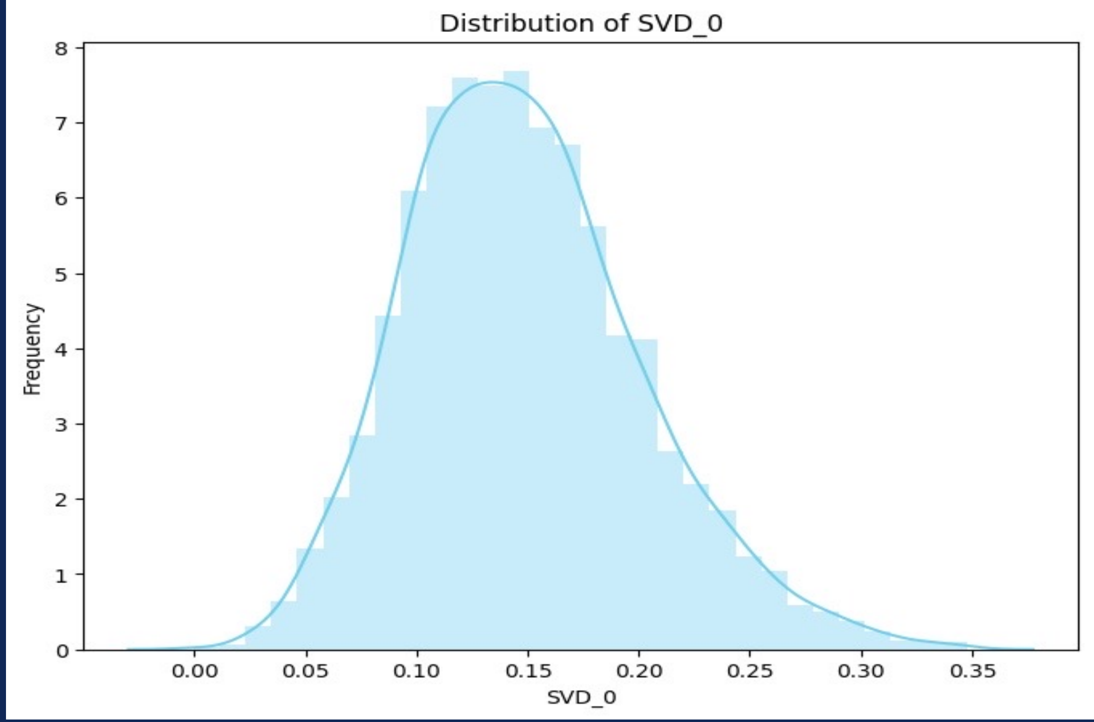
- Standardization
  Prevents sensitivity to outliers in features so we center values around the mean with unit std. deviation
- Normalization
  Common scale: scale input features to a fixed range [0,1] to ensure that no single feature disproportionately impacts the results

Standardization is useful when the features assume a normal distribution



Distribution of budget
1e−8
Frequency
budget
1e8
Distribution of Actor1_popularity
1e−5
Frequency
Actor1_popularity
Distribution of runtime
Frequency
runtime
Distribution of production_companies_populairty
1e−6
Frequency
production_companies_populairty
1e6



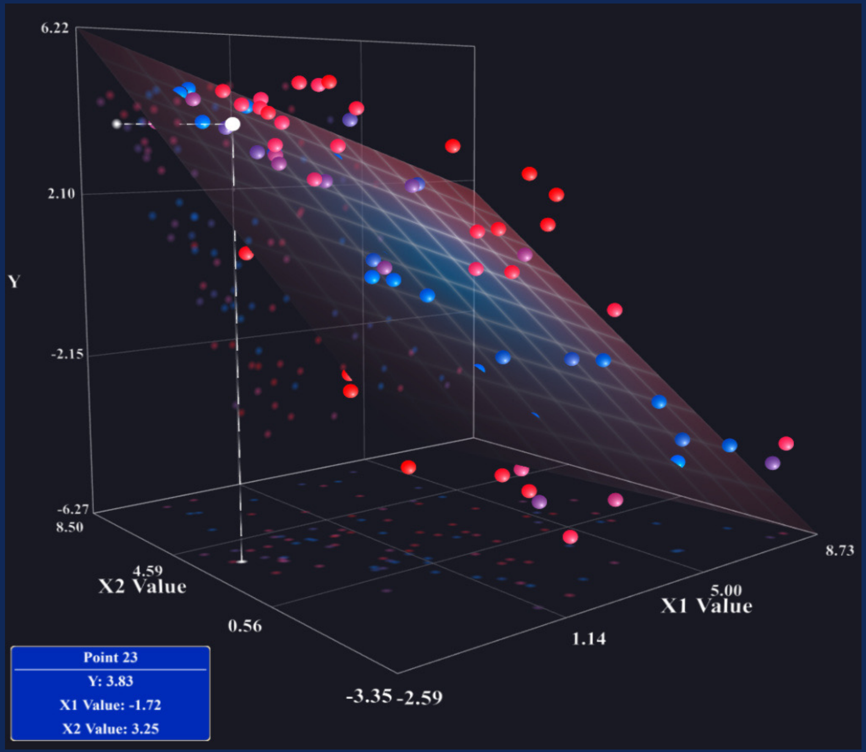


SVD_Component_1:
life: 0.1990
new: 0.1953
love: 0.1932
family: 0.1584
man: 0.1581
world: 0.1568
young: 0.1532
woman: 0.1463
story: 0.1444
film: 0.1423

SVD_Component_7:
family: 0.3762
relationship: 0.2234
new: 0.2182
father: 0.2017
world: 0.1622
school: 0.1587
war: 0.1528
mother: 0.1512
son: 0.1502
york: 0.1497

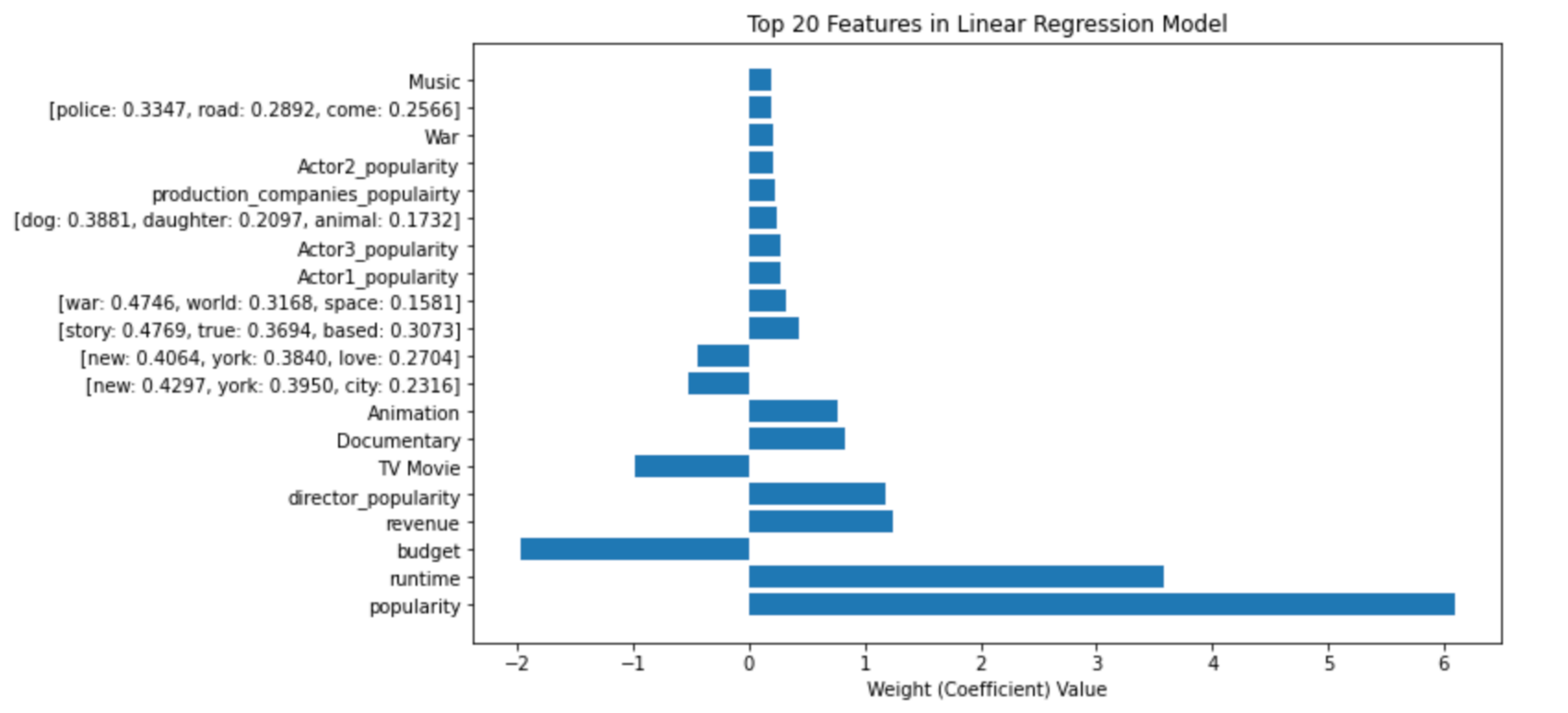
- As can be seen from the graphs, we can’t just assume normal distribution for all columns
- We apply a combination of standardization (for features with skewness < 0.5) and Normalization
- Later we also explore how applying Normalization to all columns renders different results (not a stark difference though!)

REGRESSION


6.22
2.10
Y
-2.15
-6.27
8.50
4.59
X2 Value
0.56
-3.35
-2.59
1.14
5.00
X1 Value
8.73
Point 23
Y: 3.83
X1 Value: -1.72
X2 Value: 3.25

LINEAR REGRESSION

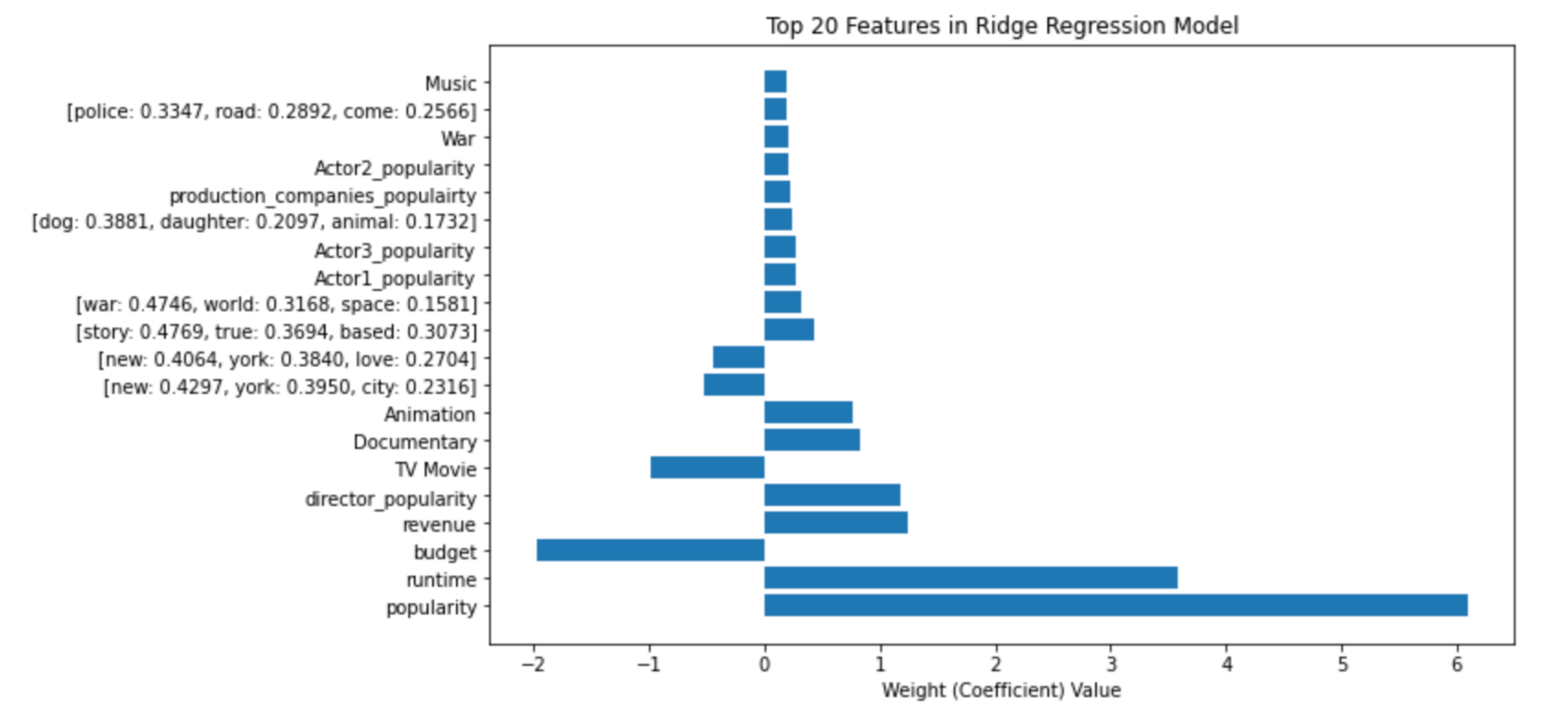
- Mean Squared Error: 1.067
- R-squared: 0.16
- Mean Absolute Error: 0.727



Top 20 Features in Linear Regression Model
Music
[police: 0.3347, road: 0.2892, come: 0.2566]
War
Actor2_popularity
production_companies_populairty
[dog: 0.3881, daughter: 0.2097, animal: 0.1732]
Actor3_popularity
Actor1_popularity
[war: 0.4746, world: 0.3168, space: 0.1581]
[story: 0.4769, true: 0.3694, based: 0.3073]
[new: 0.4064, york: 0.3840, love: 0.2704]
[new: 0.4297, york: 0.3950, city: 0.2316]
Animation
Documentary
TV Movie
director_popularity
revenue
budget
runtime
popularity
−2
−1
0
1
2
3
4
5
6
Weight (Coefficient) Value

RIDGE REGRESSION: BEST REGULARIZER

- Mean Squared Error: 1.068
- R-squared: 0.159
- Mean Absolute Error: 0.727



Top 20 Features in Ridge Regression Model
Music
[police: 0.3347, road: 0.2892, come: 0.2566]
War
Actor2_popularity
production_companies_populairty
[dog: 0.3881, daughter: 0.2097, animal: 0.1732]
Actor3_popularity
Actor1_popularity
[war: 0.4746, world: 0.3168, space: 0.1581]
[story: 0.4769, true: 0.3694, based: 0.3073]
[new: 0.4064, york: 0.3840, love: 0.2704]
[new: 0.4297, york: 0.3950, city: 0.2316]
Animation
Documentary
TV Movie
director_popularity
revenue
budget
runtime
popularity
−2
−1
0
1
2
3
4
5
6
Weight (Coefficient) Value

SVR: SUPPORT VECTOR REGRESSION

Parameters tried c = [0.1, 1, 5] with Degree = 2,3,4.

- Linear SVR for c = 1
- Squared Error: 1.028
- R-squared: 0.19
- Mean Absolute Error: 0.68

Polynomial SVR for c=1 and Degree = 2

- Mean Squared Error: 1.25
- R-squared: 0.014
- Mean Absolute Error: 0.775

RBF for c=5

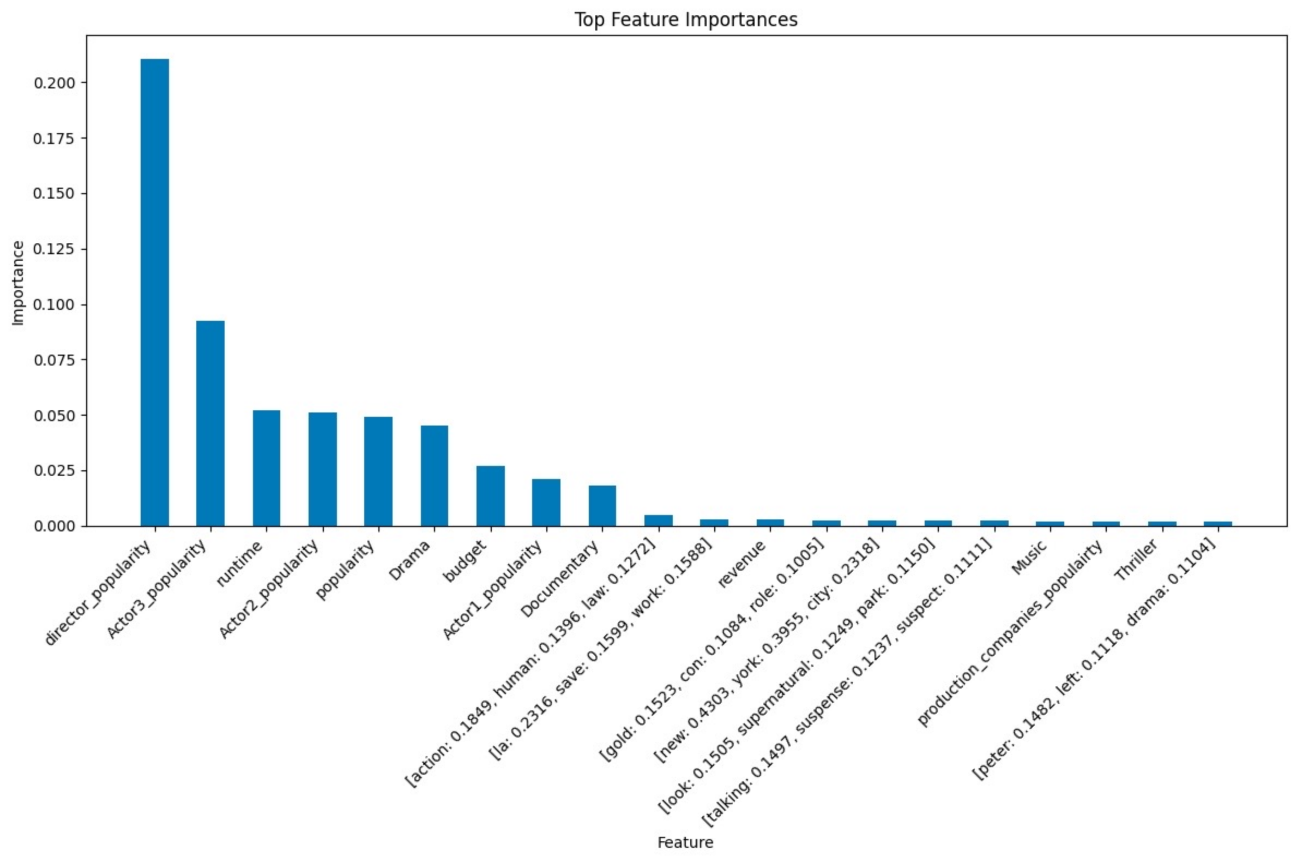
- Mean Squared Error: 1.15
- R-squared: 0.087
- Mean Absolute Error: 0.753

RANDOM FOREST

Mean Squared Error: 0.53

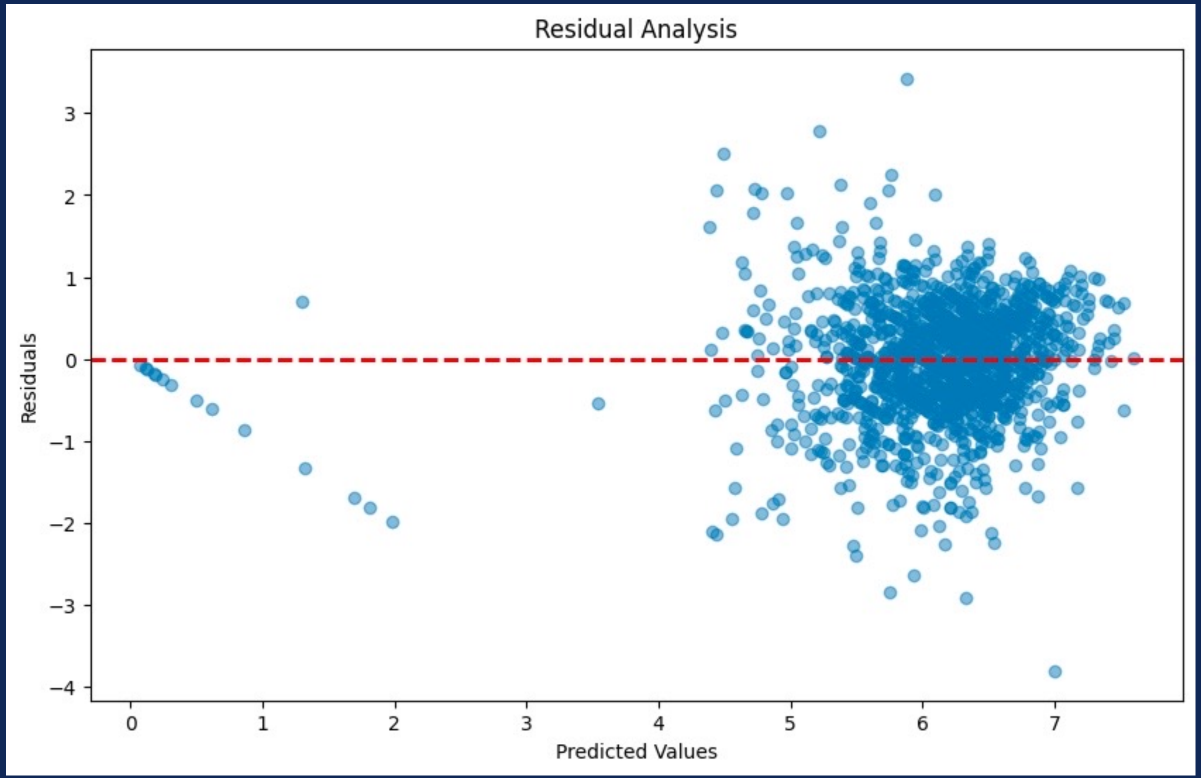
R-squared: 0.58

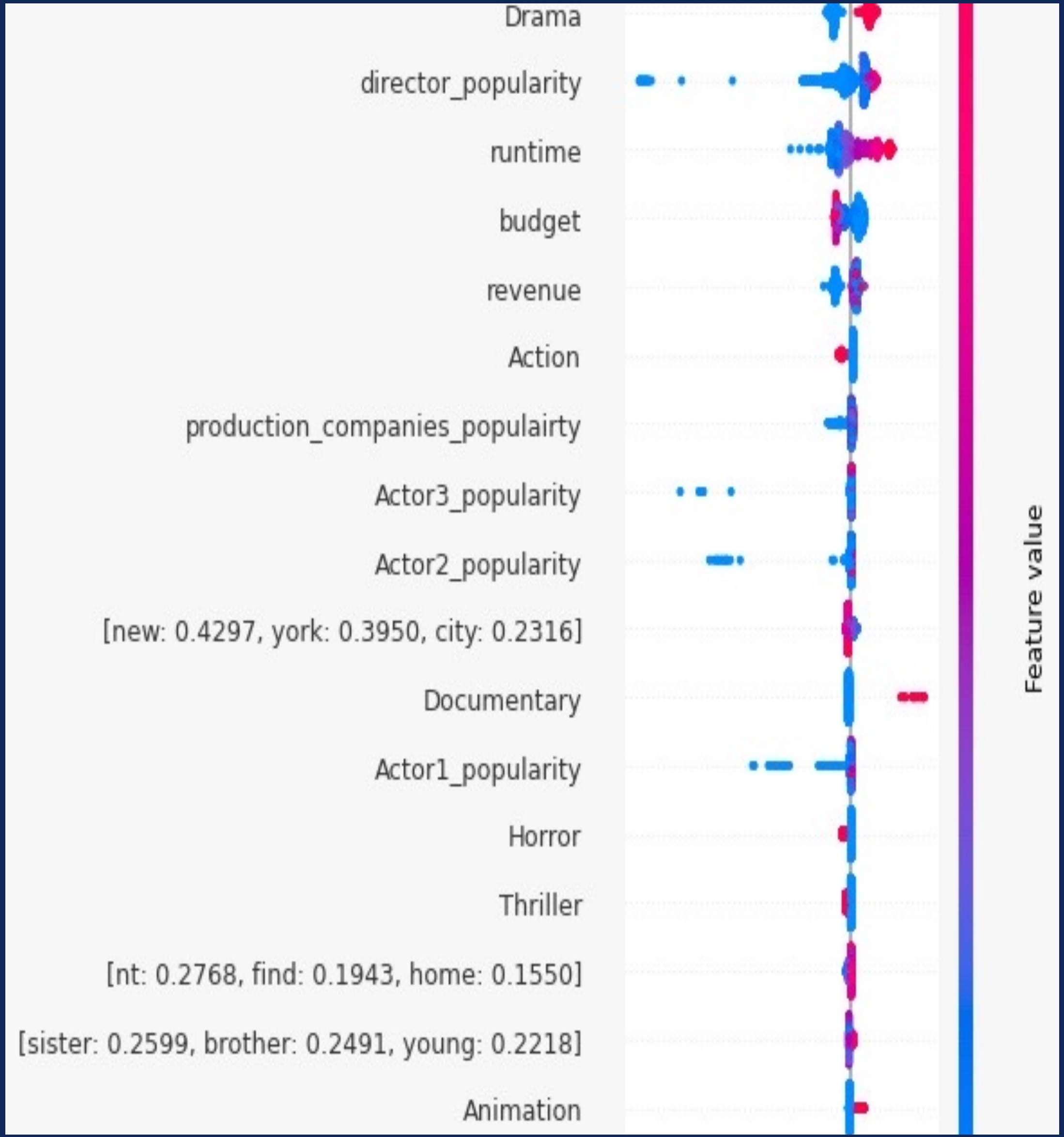
Mean Absolute Error: 0.558

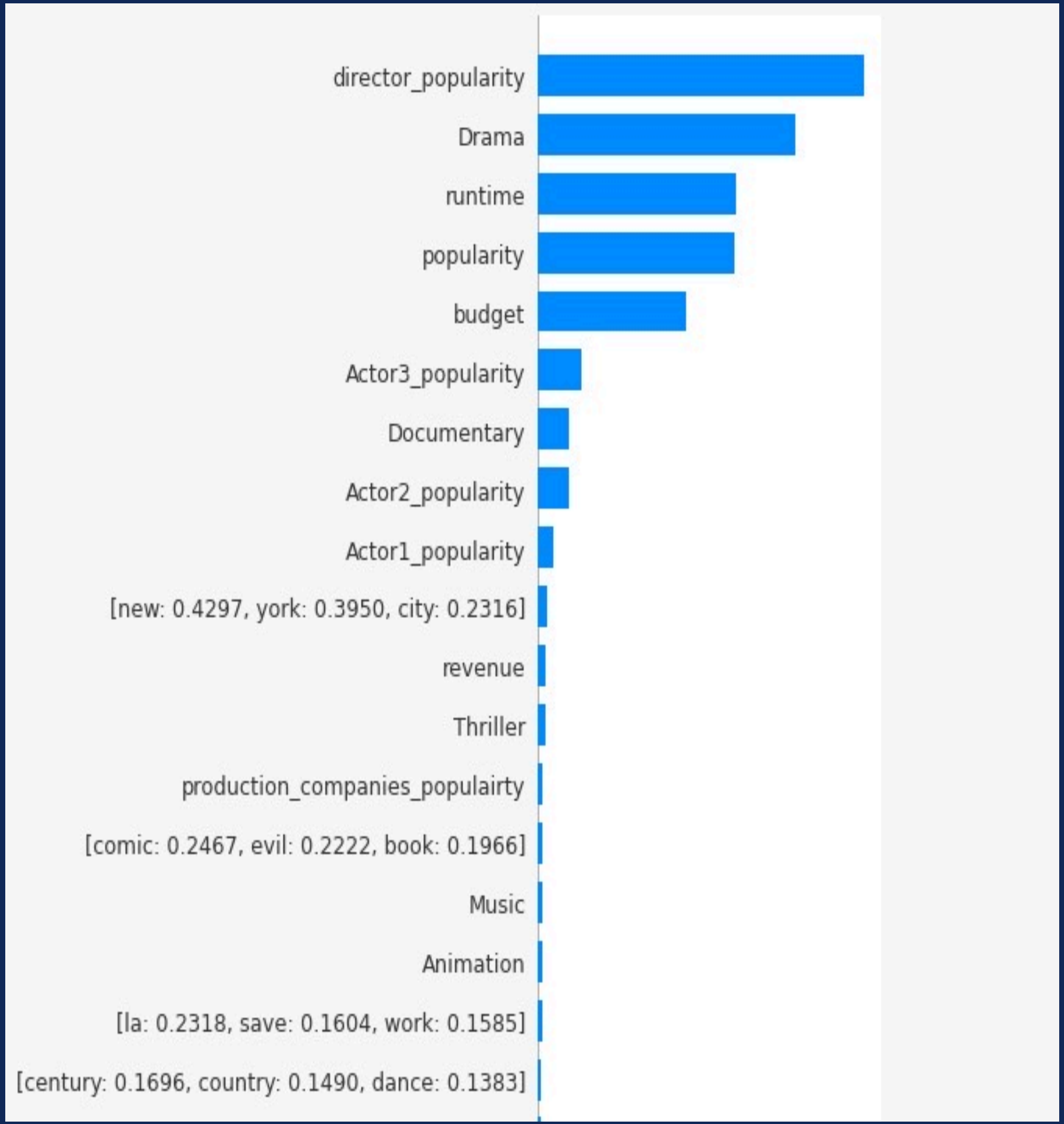


RANDOM FOREST

- Used cross-validation folds=5, saw better and stable results with estimators=50.
- Mean MSE: 0.57
- Standard Deviation of MSE: 0.022

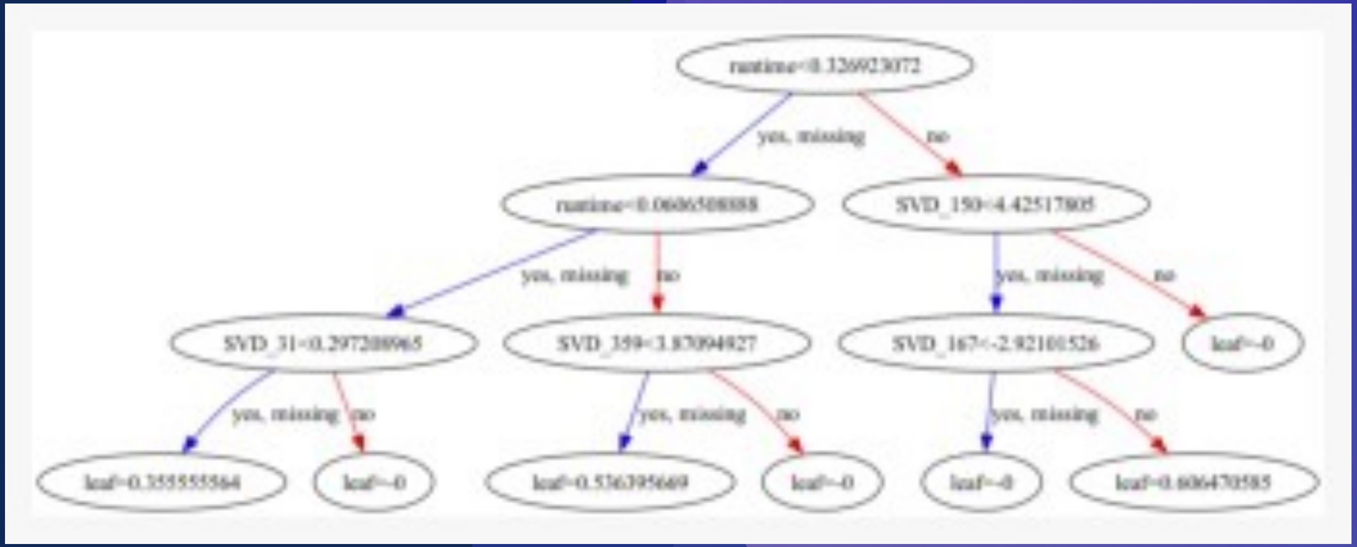



Drama
director_popularity
runtime
budget
revenue
Action
production_companies_populairty
Actor3_popularity
Actor2_popularity
[new: 0.4297, york: 0.3950, city: 0.2316]
Documentary
Actor1_popularity
Horror
Thriller
[nt: 0.2768, find: 0.1943, home: 0.1550]
[sister: 0.2599, brother: 0.2491, young: 0.2218]
Animation
Feature value


director_popularity
Drama
runtime
popularity
budget
Actor3_popularity
Documentary
Actor2_popularity
Actor1_popularity
[new: 0.4297, york: 0.3950, city: 0.2316]
revenue
Thriller
production_companies_populairty
[comic: 0.2467, evil: 0.2222, book: 0.1966]
Music
Animation
[la: 0.2318, save: 0.1604, work: 0.1585]
[century: 0.1696, country: 0.1490, dance: 0.1383]

XGBOOST

- Mean Squared Error: 0.729
- R-squared: 0.425
- Mean Absolute Error: 0.627

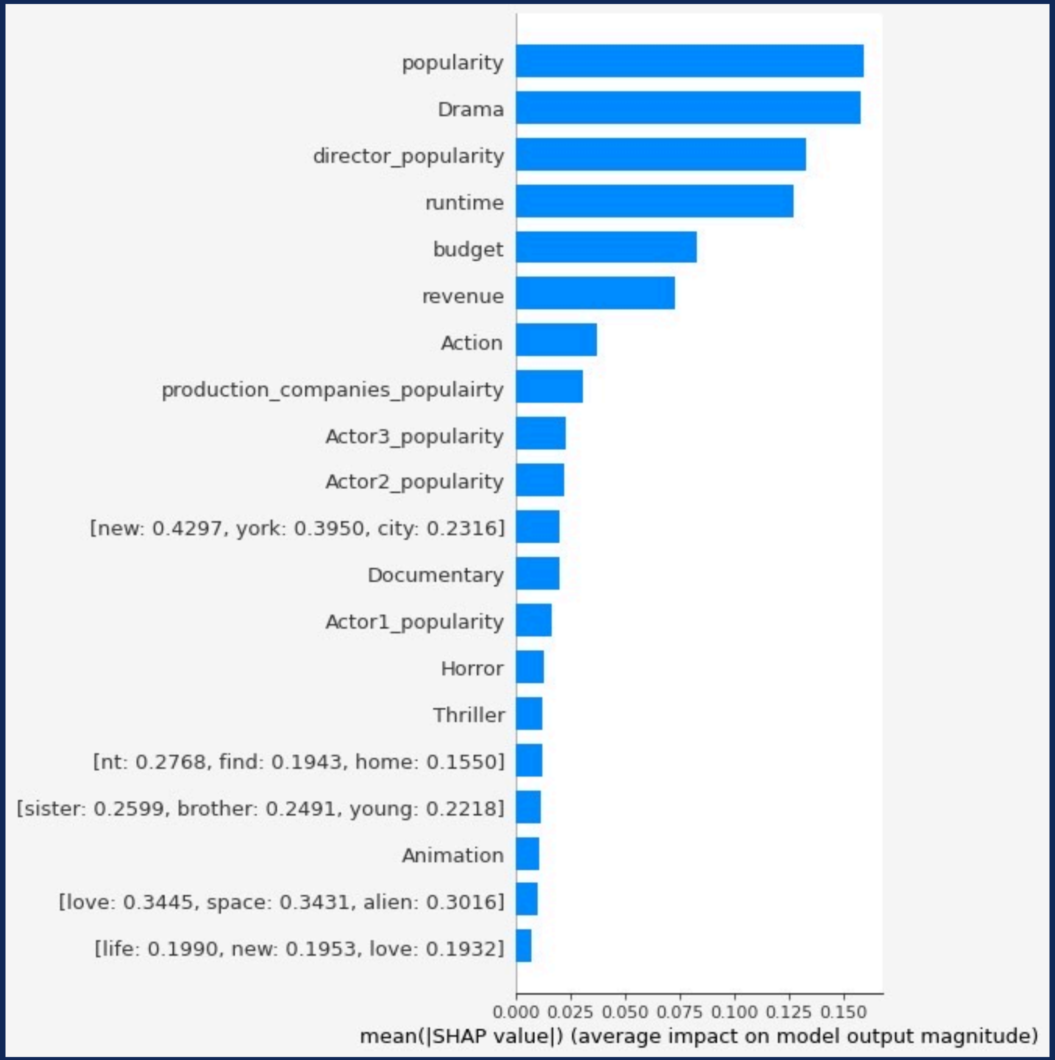


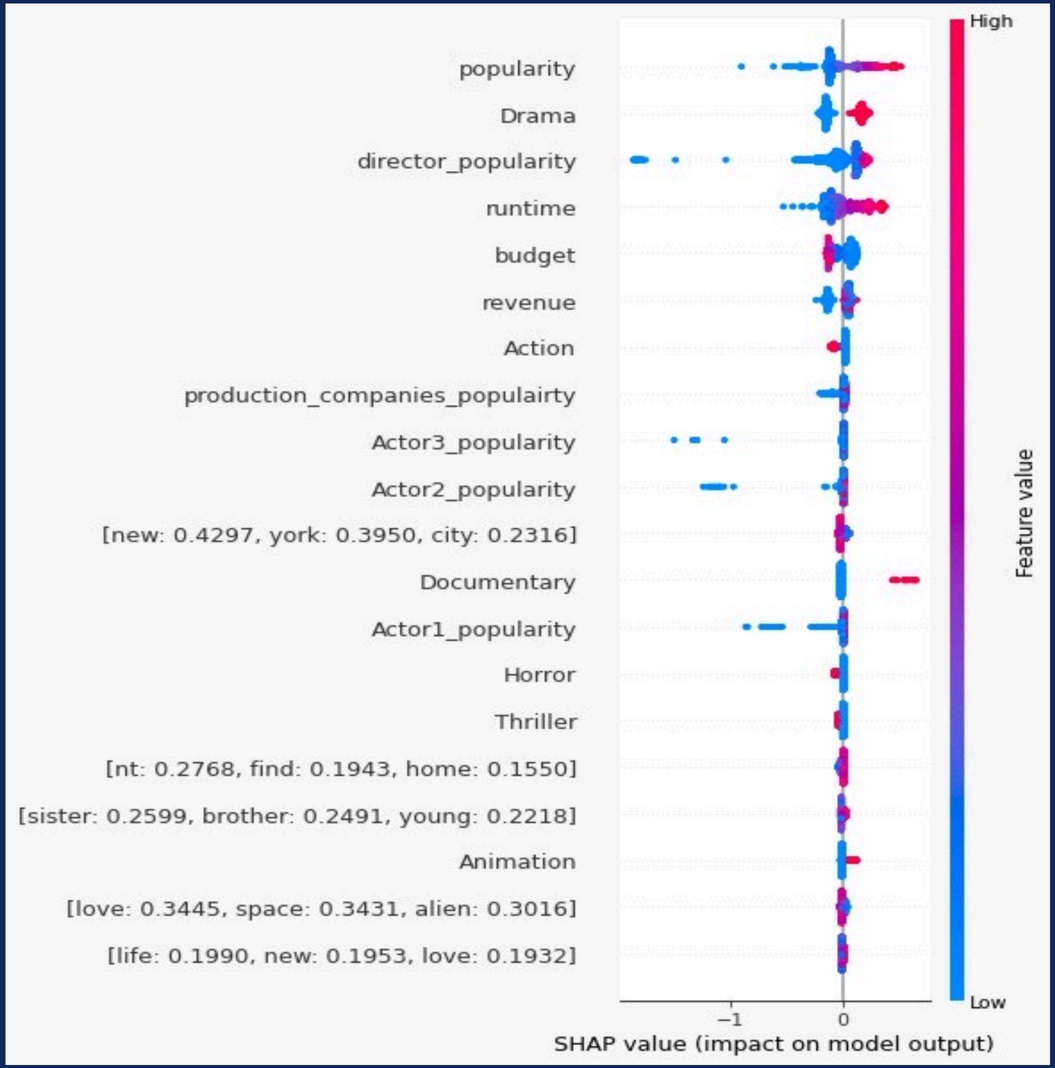
For hyperparameter tuning, used GridSearchCV:

param_grid = { 'learning_rate': [0.1, 0.4, 0.5, 1.0],
'n_estimators': [10, 30, 50]
, 'max_depth': [3, 4, 8]}

Best Parameters:
Learning_rate: 0.1
Max_depth: 3
n_estimators: 50
Mean Squared Error (Best Model): 0.5412652381261654
R-squared: 0.5739069215486285
Mean Absolute Error: 0.5679970501509389

XGBOOST


popularity
Drama
director_popularity
runtime
budget
revenue
Action
production_companies_populairty
Actor3_popularity
Actor2_popularity
[new: 0.4297, york: 0.3950, city: 0.2316]
Documentary
Actor1_popularity
Horror
Thriller
[nt: 0.2768, find: 0.1943, home: 0.1550]
[sister: 0.2599, brother: 0.2491, young: 0.2218]
Animation
[love: 0.3445, space: 0.3431, alien: 0.3016]
[life: 0.1990, new: 0.1953, love: 0.1932]
0.000 0.025 0.050 0.075 0.100 0.125 0.150
mean(|SHAP value|) (average impact on model output magnitude)


High
popularity
Drama
director_popularity
runtime
budget
revenue
Action
production_companies_populairty
Actor3_popularity
Actor2_popularity
[new: 0.4297, york: 0.3950, city: 0.2316]
Documentary
Actor1_popularity
Horror
Thriller
[nt: 0.2768, find: 0.1943, home: 0.1550]
[sister: 0.2599, brother: 0.2491, young: 0.2218]
Animation
[love: 0.3445, space: 0.3431, alien: 0.3016]
[life: 0.1990, new: 0.1953, love: 0.1932]
Feature value
Low
−1
0
SHAP value (impact on model output)

SUMMARY

- Random Forest produces the best results on the data as the error is relatively less and the R^2 value is higher than the other regressions.
- Insights:
  - Drama: If a movie is a Drama (a sub-genre), it is likely to have better ratings
  - Director popularity: As can be seen, if a director is unpopular, the movie gets lesser ratings, and it is unusual for a bad director to have a good rated movie
  - Runtime: Longer the movie, better the ratings

SUMMARY

- More Insights
  - Budget: Surprisingly, low budget movies have done well and a small fraction of high budget films have not done well enough
  - Actor popularity: Actor_1 will obviously be popular (mostly) but if Actor_3 is unpopular, the movie usually gets rated low that means having 3 popular actors performs better than less than 3
  - Keywords in plots like: New York, Love, Comic, Evil, Alien, Force positively impact ratings
  - Lastly, some genres like Thriller, Action, Documentary also play important roles in ratings.

THANK YOU

Anuva Sehgal

Ravan Buddha

MACHINE LEARNING FINAL PROJECT

Predicting Startup Outcomes: Operating, Acquired, or Closed

Presented by Ian Liao

Problem Statement

- Startup: a company that is in their first stages of operations. 90% of them fail due to bad product market fit, marketing problems, team problems or other issues, mostly within the first few years.
- Startup investment can be very risky due to the high failure rate of startups, especially for angel investors and venture capitalists.
- This project aims to find the important features that lead to startup success and forecast a company's success with supervised machine learning methods.

Methodology

- Data Preprocessing
  - Multiple Dataset
- Feature Engineering
  - One-hot encoding
- Class Imbalance
  - SMOTE (oversample minority class)
- Model Training:
  - Decision Tree
  - Random Forest
  - SVM
  - XGBoost
  - LightGBM

Classification Accuracy

After Handling class imbalance issue, we trained and tested data with different approaches

METHOD	PRECISION	RECALL	ACCURACY
Decision Tree	0.87	0.84	0.84
Random Forest	0.88	0.86	0.86
Gradient Boosting	0.89	0.86	0.86
SVM	0.79	0.75	0.75
XGBoost	0.89	0.88	0.88
LightGBM	0.89	0.88	0.88

Conclusion:

Crucial Features to Determine the Success of a Startup

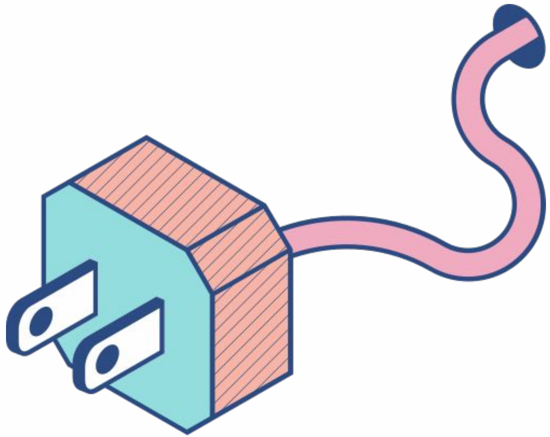
- Total Funding
- Seed Round Funding
- Found to Fund Time Period

Future Improvements

Although the classification algorithm provides a satisfactory result, the prediction could be more powerful and applicable with following improvements:

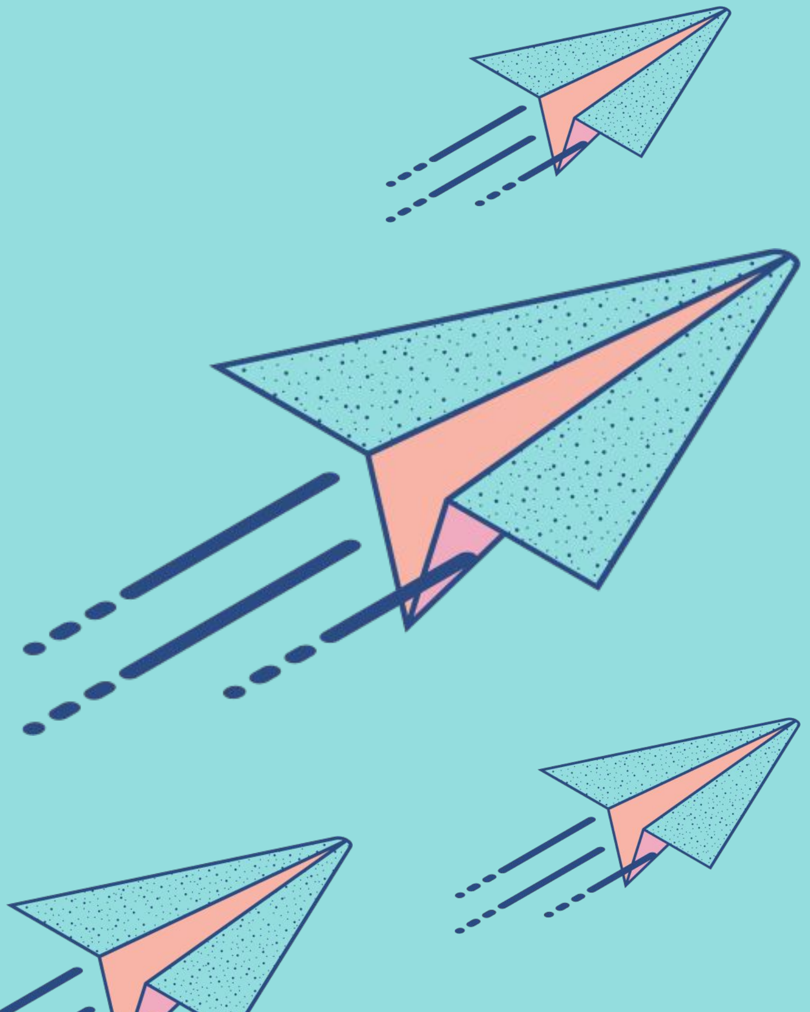
General Solution: CRISP-DM

1. Keep Update Data (After Covid, AI, ...)
2. Based on prediction performance, keep adjusting model
3. Deployment



Do you have any questions?

It is Q&A time!



STOCK PRICE PREDICTION

Predicting short-term price movements in the Nasdaq Stock Exchange closing auction

Jinseok (Jake) Yoon

Ari Khaytser

Aavishkar Gautam

INTRODUCTION

- The study takes an in-depth look at the Nasdaq Closing Cross, a key event in financial markets for setting the official final prices of securities, crucial for accurate market closing.

- The closing price is important due to its significant impact on portfolio valuations and market sentiment.

- The task is complicated by factors like market volatility, high volume of trades, rapid shifts in investor sentiment, information asymmetry, and the impact of strategic moves by large investors, all converging in the market's final moments.

- We aim to provide a clearer understanding of order book behavior and auction pricing strategies, and understand the nature of stock pricing.

Dataset

- Stock and Date Identifiers: Unique identifiers for each stock (stock_id) and the date of trading.
- Imbalance Size and Direction: Quantifies unmatched trade volume at reference prices, with flags indicating buy or sell imbalances.
- Reference and Crossing Prices: The optimized price points for trade matching, considering auction and continuous market orders.
- Bid/Ask Prices and Sizes: Price and quantity information of bidding and asking orders.
- Weighted Average Price (WAP): Weighted average price of non-auction book orders.
- Target Metrics: 60 seconds future WAP & price index movements for the prediction.
  (*Training set only - this is the metric that the models is trained on).

Methodology

ARIMA Model

- Predicts Time Series Data: Ideal for forecasting future data points in time series with trends.
- Handles Non-Stationarity: Effective in dealing with data where the mean changes over time.
- Seasonal Adjustment: Uses seasonal differencing to manage data with periodic changes.

Methodology

FT-Transformer Model

- Time Series Analysis for Stock Prices: Specialized in understanding patterns in data over a 10-minute window.
- Adapts NLP Techniques: Uses methods from Natural Language Processing to interpret structured tabular data.
- Context Understanding: Effective in deducing meanings from past data to predict future stock prices.

Results

Results were evaluated on the Mean Absolute Error (MAE) between the predicted return and the observed target.

$$\text{MAE} = \frac{1}{n} \Sigma |y_i - x_i|$$

where,

n is the total number of data points.
y_i is the predicted value for data point i.
x_i is the observed value for data point i.

FT-transformer - **5.3** MAE

ARIMA Model - 5.8 MAE for general model
- 4.0 MAE after parameter search for each individual stock (fine-tuned parameters for each stock)

Conclusion and Limitations

- ARIMA's Limitations: Better in certain conditions due to its simplicity, but unable to learn complex relationships of the market variables (high bias).

- Passive Reaction to Market Changes: ARIMA is based on moving averages and does not do well at predicting abrupt market shifts.

- Transformer's Advantage: Can proactively learn and predict sudden changes, unlike ARIMA, which only responds passively to market movements. (for example, sudden cancellation of a large block order and its impact on the impending auction price)

Stock Price Prediction through Regression Algorithms

By David Chen
Group number 17

Introduction

- Stocks are portions of a company that are bought, sold, and traded on a public stock exchange.
- Each stock represents a share of a company.
- Many factors can determine the increase or decrease of stock price
- My Project aims to determine the closing price of a company's stock using linear regression and time-series analysis algorithms

Relevance

- Stock prices are used to determine the shape and condition of the economy and a specific company[2]
- Given their high volatility, it can be significant for investors to predict stock prices accurately in terms of financial returns.

Algorithms used

- Linear Regression: an algorithm that takes in one or more inputs against a single-variable output using a linear equation
- Time-series: An algorithm that plots the results or output with respect to time

Methodology

- First use datasets from various companies
    - Microsoft
    - Apple
- Graphed them in a time-series
- Divided the dataset into X(input) and y(output) with training and testing sets
- Compared the actual time-series graphs with the linear regression model comparison

Previous Attempts

- Many researchers, economists and computer scientists have derived methods to calculate and predict stock prices
- One researcher used the Highest, lowest, and opening price to calculate the closing price[1]
- Another used Logistic regression using the same input features to predict the latter output[2]

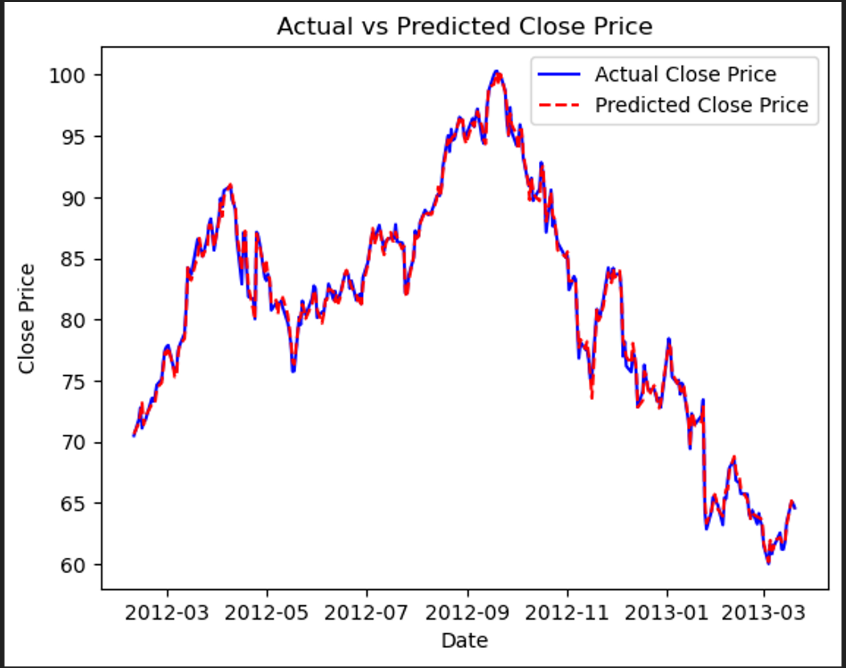
Dataset

- I used multiple datasets from multiple companies(Microsoft, Apple)
- I compared the closing price with respect to the opening price of each day for each company
- Finally, I compare the graphs of the predicted vs actual outcome along with their statistics such as l2-loss, mean-squared error, and mean absolute error

Results(Apple)

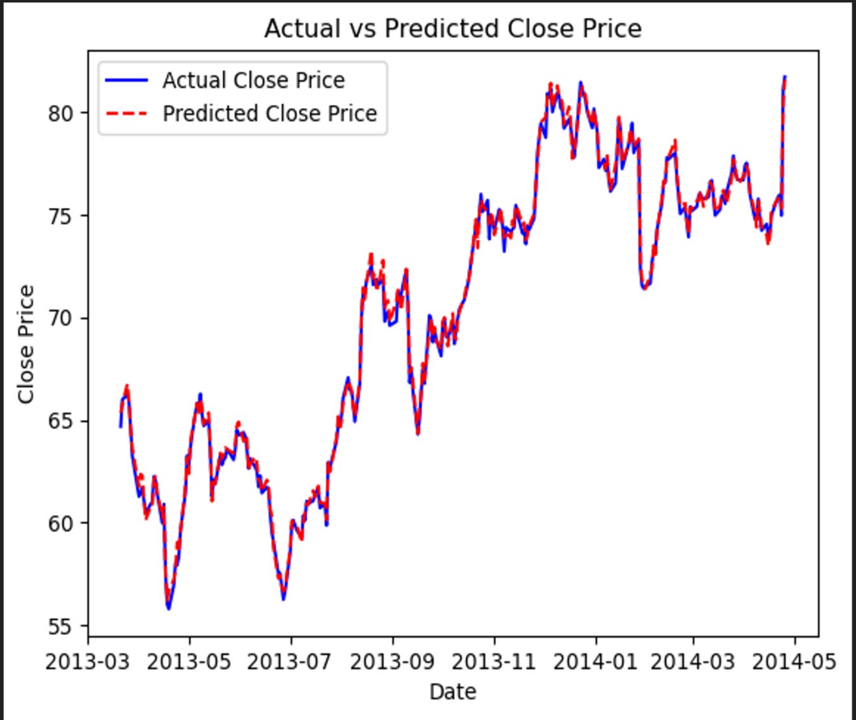
Apple's Information:

- Mean Absolute Error: 0.3910088508193302
- Mean Squared Error: 0.2533723609410882
- Root Mean Squared Error: 0.5033610641886083



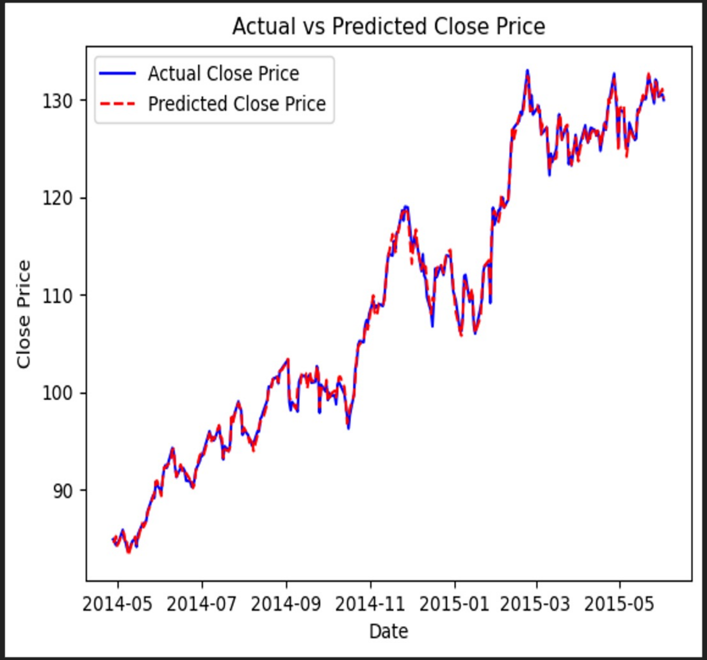
Results(Apple)

- Mean Absolute Error: 0.26338358803048345
- Mean Squared Error: 0.11411611583777917
- Root Mean Squared Error: 0.33781076927442555



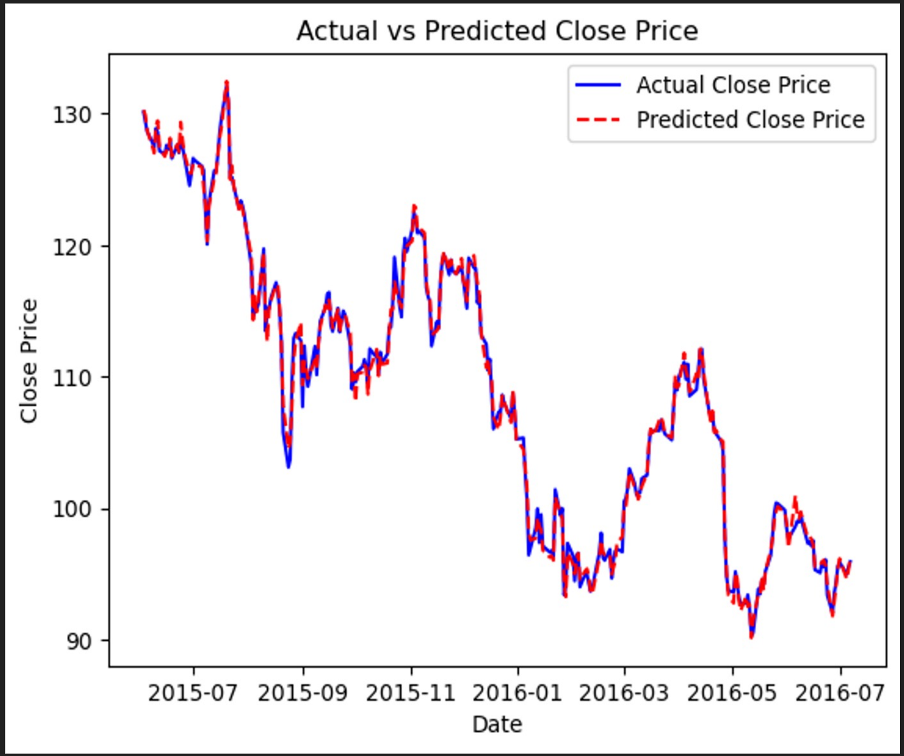
Results(Apple)

- Mean Absolute Error: 0.4040643298607208
- Mean Squared Error: 0.31285079490611245
- Root Mean Squared Error: 0.5593306668743566



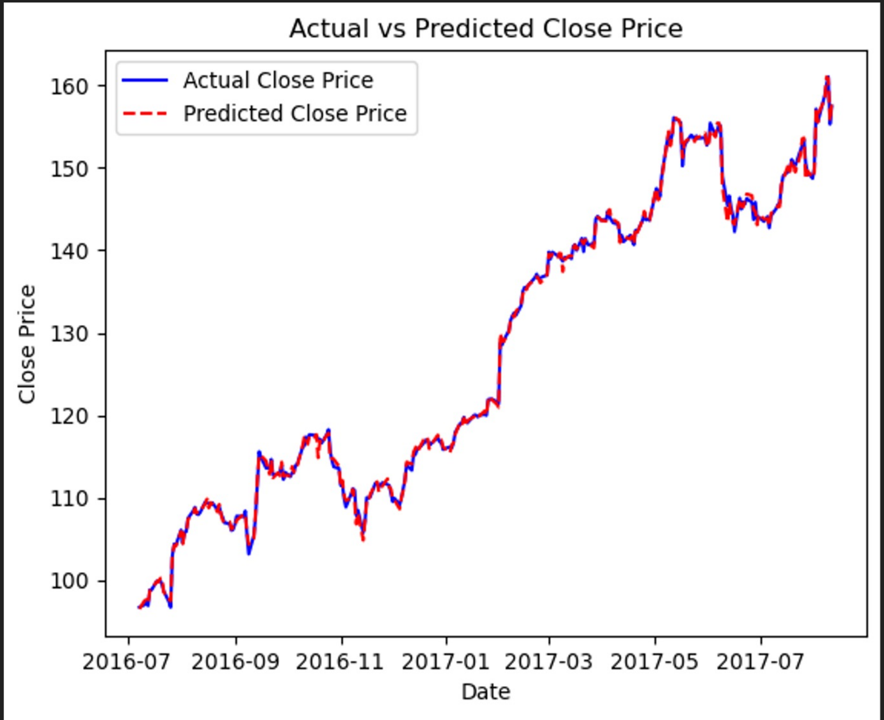
Results(Apple)

- Mean Absolute Error: 0.4951268146439754
- Mean Squared Error: 0.432099364410951
- Root Mean Squared Error: 0.6573426537285946



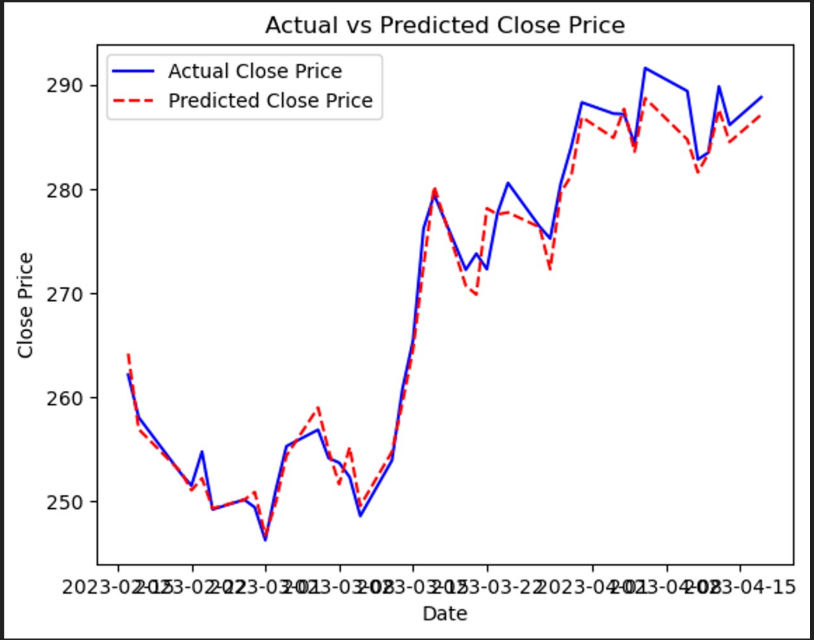
Results(Apple)

- Mean Absolute Error: 0.36065284032134914
- Mean Squared Error: 0.23869596158030154
- Root Mean Squared Error: 0.4885652070914398



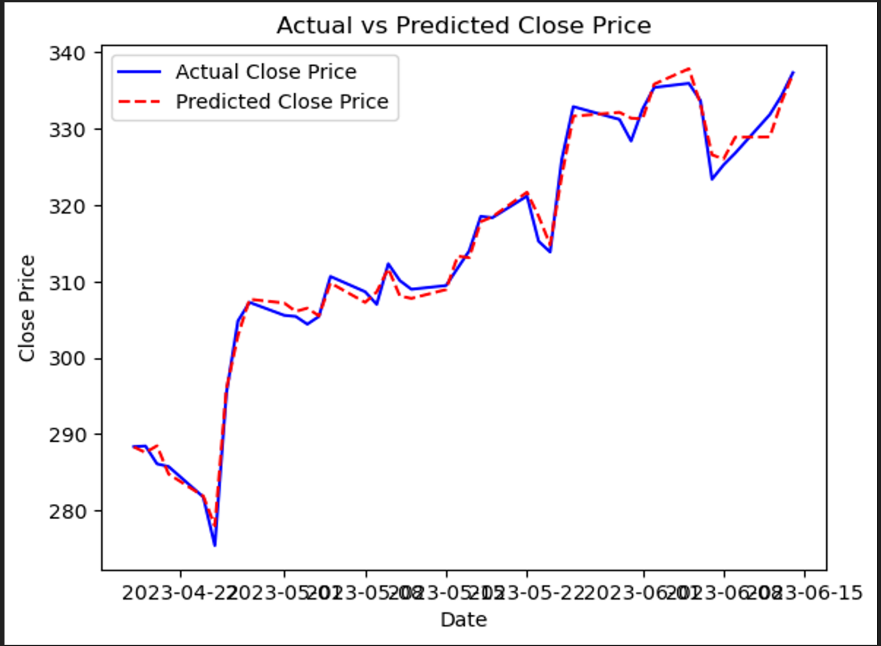
Results(Microsoft)

- Mean Absolute Error: 1.633001243677619
- Mean Squared Error: 4.368625276789374
- Root Mean Squared Error: 2.0901256605260303



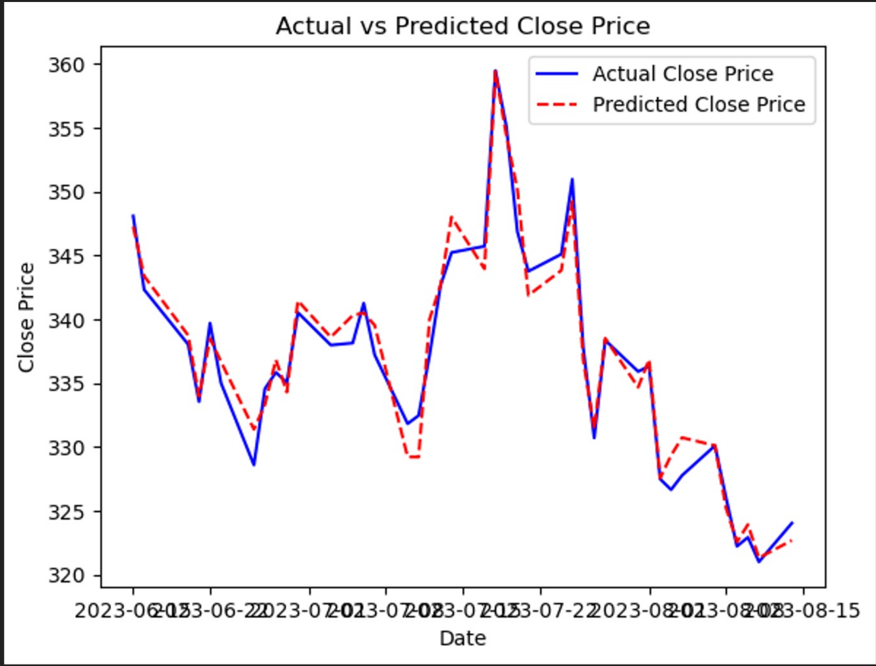
Results(Microsoft)

- Mean Absolute Error: 1.2655330039990822
- Mean Squared Error: 2.410999727683625
- Root Mean Squared Error: 1.552739426846509



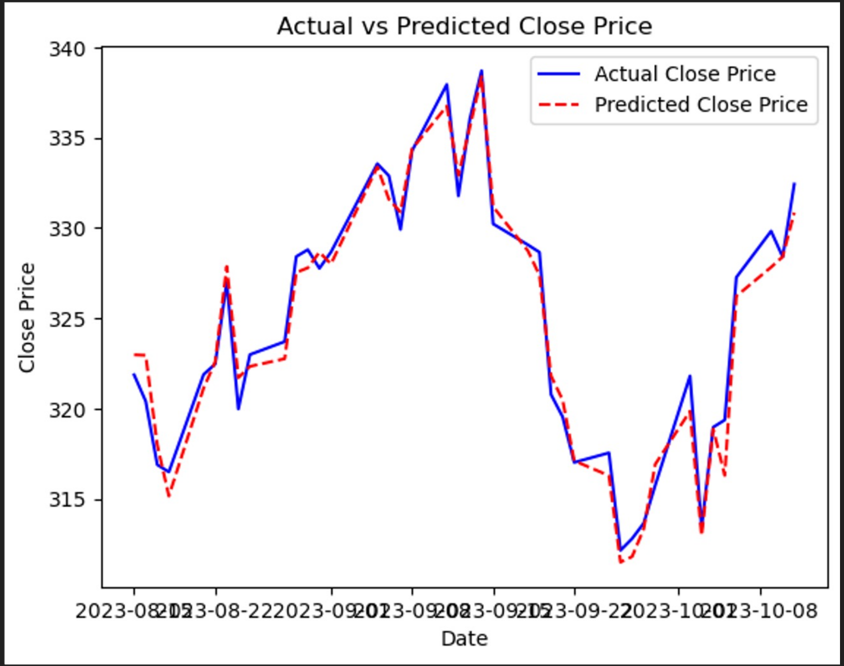
Results(Microsoft)

- Mean Absolute Error: 1.3125163672426126
- Mean Squared Error: 2.638760461741421
- Root Mean Squared Error: 1.6244261946119378



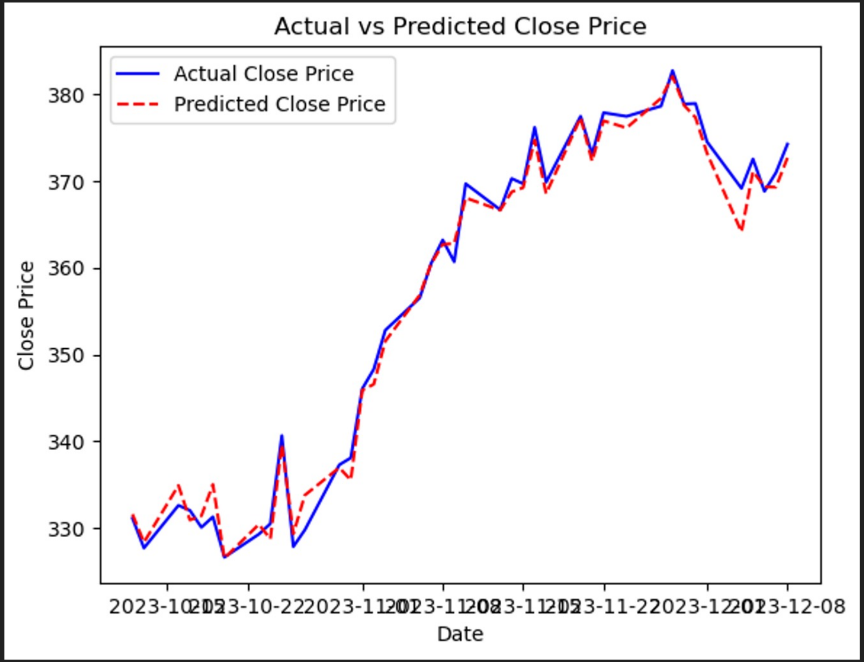
Results(Microsoft)

- Mean Absolute Error: 0.9614522490878574
- Mean Squared Error: 1.3432758267900222
- Root Mean Squared Error: 1.158997768242037



Results(Microsoft)

- Mean Absolute Error: 1.2831176813179357
- Mean Squared Error: 2.762213194763548
- Root Mean Squared Error: 1.6619907324541698



Conclusion

- Linear regression can be a helpful tool to predict stock prices when using long-term data from many years
- When using short-term data, there can be less accuracy with stock price prediction and higher error
- Linear regression tends to be less accurate in sudden changes of prices
- The regressive problem type and linear-nature of stock prices make linear regression an ideal algorithm for long-term stock price prediction

References

1.Antad, Sonali, et al. "Stock Price Prediction Website Using Linear Regression - A Machine Learning Algorithm." ITM Web of Conferences, vol. 56, 2023.

2.Rishi, Taran (2022) "Stock Market Analysis Using Linear Regression," Proceedings of the Jepson Undergraduate Conference on International Economics: Vol. 4, Article 4. Available at: https://scholarworks.uni.edu/jucie/vol4/iss1/4

3. Selvaraj, Rohini. "Stock Price Prediction of Apple Inc." Kaggle, https://www.kaggle.com/datasets/rohiniselvaraj0107/stock-price-prediction-of-apple-inc.

4."Microsoft Corporation (MSFT) - Yahoo Finance." Yahoo Finance, https://finance.yahoo.com/quote/MSFT/history.

NYU

Final Course Project

Stock Return Prediction Using LSTM and Technical Analysis Indicators

PRESENTED BY SUNNY YANG
12/12/2023
GROUP 19

PART 01

Introduction

PART 01

Introduction

Stock prediction

The stock market, characterized by its dynamic and complex nature, presents a significant challenge for investors aiming to predict market movements and make profitable trading decisions. Technical analysis, employing a range of indicators to analyze market trends and forecast future price movements, has long been a staple in the trader's toolkit.

LSTM

The prediction of stock prices is a challenging task due to the inherent complexity and volatility of financial markets. Traditional methods often fail to capture the intricate patterns and dependencies present in stock price data. However, LSTM models have shown great potential in capturing temporal dependencies and making accurate predictions in various time series forecasting tasks.

PART 02

Dataset and LSTM

Dataset

Date	Open	High	Low	Close	Adj Close	Volume	Return
2016-01-04	54.320000	54.799999	53.389999	54.799999	48.698887	53778000	NaN
2016-01-05	54.930000	55.389999	54.540001	55.049999	48.921055	34079700	0.456207
2016-01-06	54.320000	54.400002	53.639999	54.049999	48.032387	39518900	-1.816535
2016-01-07	52.700001	53.490002	52.070000	52.169998	46.361691	56564900	-3.478270
2016-01-08	52.369999	53.279999	52.150002	52.330002	46.503880	48754000	0.306695
...	...	...	...	...	...	...	...
2023-12-05	366.450012	373.079987	365.619995	372.519989	372.519989	23065000	0.915635
2023-12-06	373.540009	374.179993	368.029999	368.799988	368.799988	21182100	-0.998604
2023-12-07	368.230011	371.450012	366.320007	370.950012	370.950012	23118900	0.582978
2023-12-08	369.200012	374.459991	368.230011	374.230011	374.230011	20144800	0.884216
2023-12-11	368.480011	371.600006	366.100006	371.299988	371.299988	27686500	-0.782947

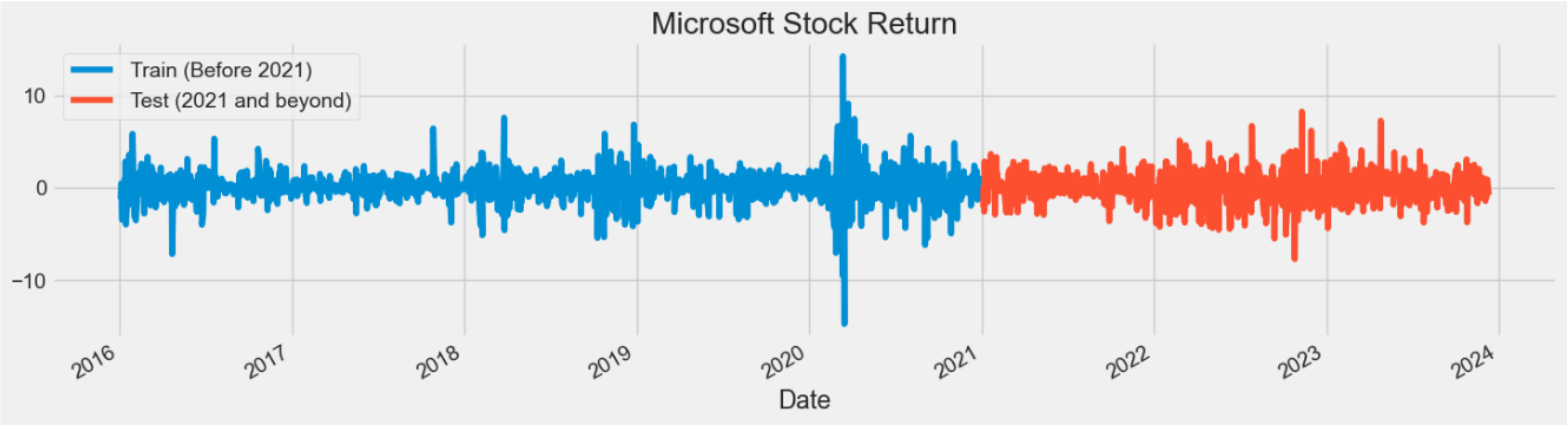
1999 rows × 7 columns

Stock: Microsoft(MSFT)

Train: 2016-2020

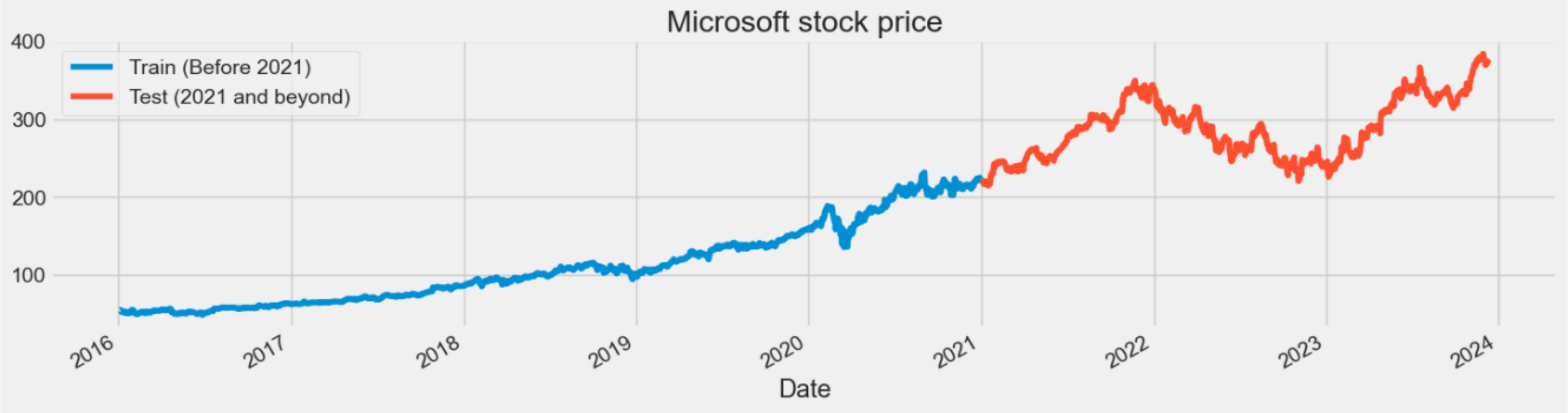
Test: 2021-present

Dataset


Microsoft Stock Return
Train (Before 2021)
Test (2021 and beyond)
10
0
−10
2016
2017
2018
2019
2020
2021
2022
2023
2024
Date

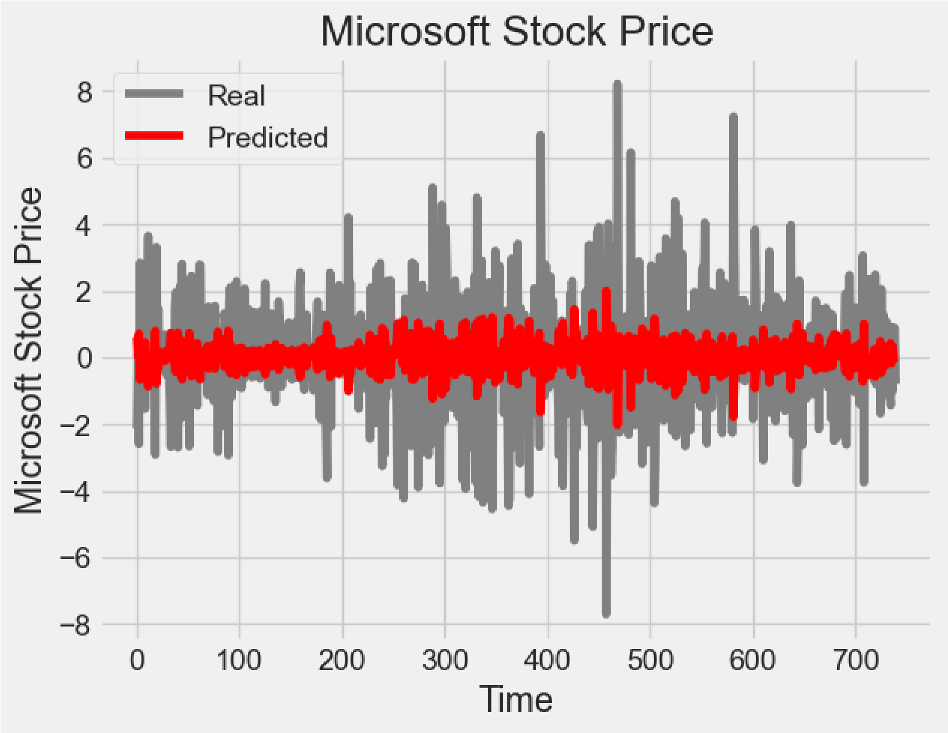
NYU

Dataset


Microsoft stock price
Train (Before 2021)
Test (2021 and beyond)
400
300
200
100
2016
2017
2018
2019
2020
2021
2022
2023
2024
Date

NYU

LSTM

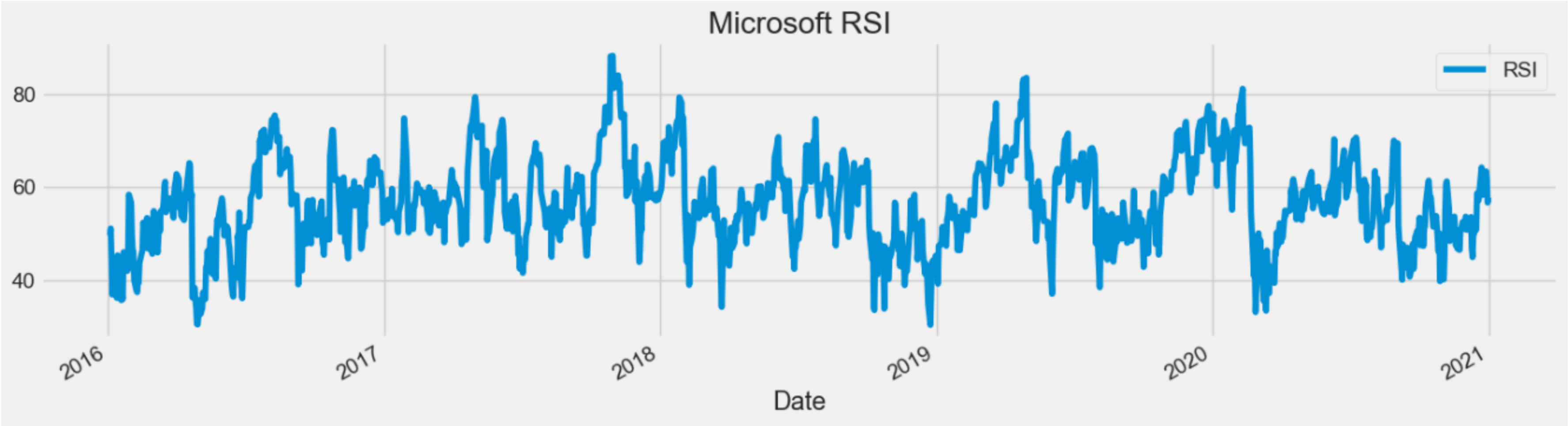

Microsoft Stock Price
Real
Predicted
Microsoft Stock Price
8
6
4
2
0
−2
−4
−6
−8
0
100
200
300
400
500
600
700
Time

NYU

PART 03

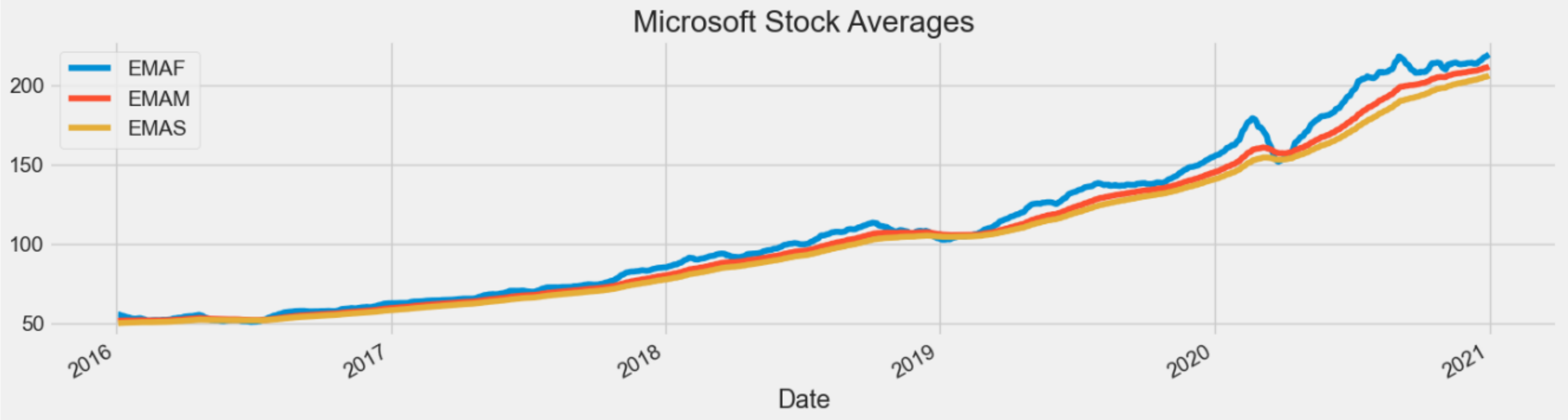
Features and Result

Features RSI



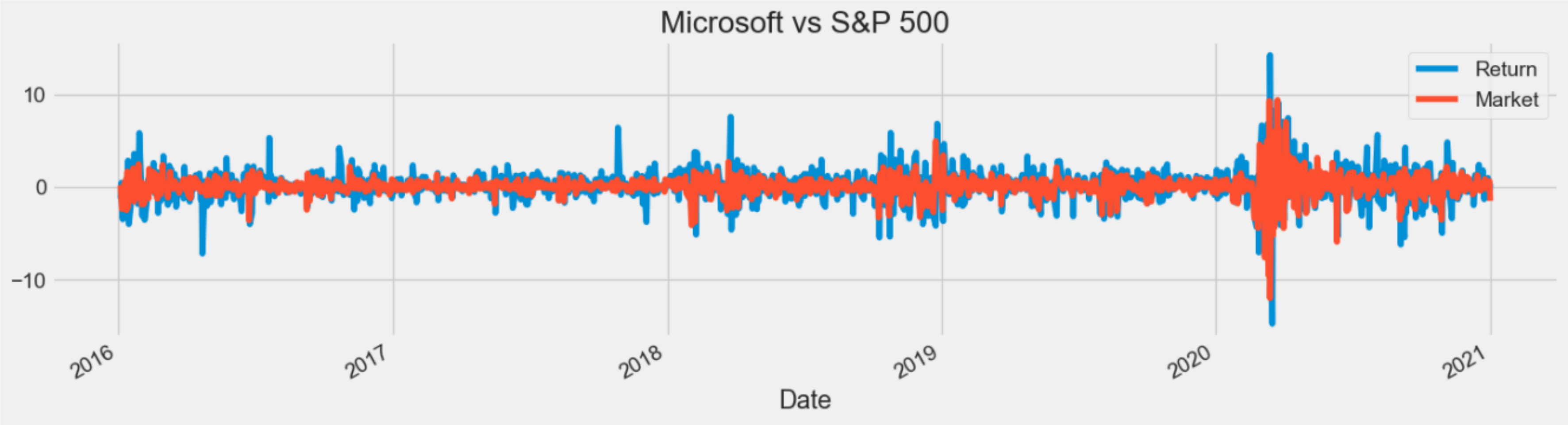
Relative Strength Index (RSI): is a momentum indicator used in technical analysis. RSI measures the speed and magnitude of a security's recent price changes to evaluate overvalued or undervalued conditions in the price of that security.

Features EMA

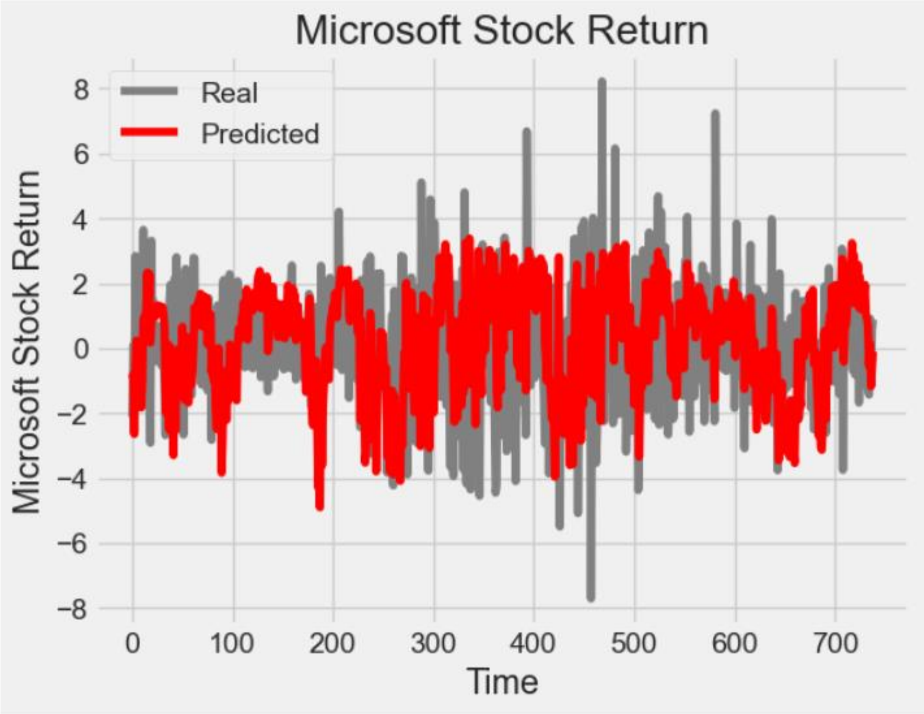


Exponential Moving Average (EMA): is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average. An exponentially weighted moving average reacts more significantly to recent price changes than a simple moving average simple moving average (SMA), which applies an equal weight to all observations in the period. Traders often use several different EMA lengths, such as 10-day, 50-day, and 200-day moving averages.

Features Market



Result

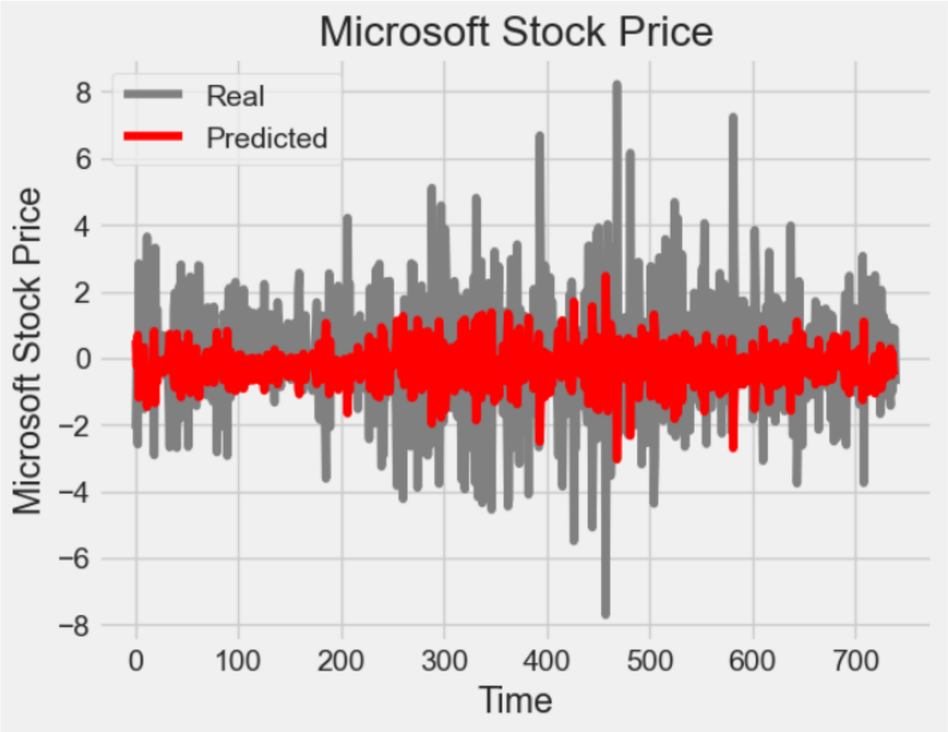

Microsoft Stock Return
Real
Predicted
Microsoft Stock Return
8
6
4
2
0
−2
−4
−6
−8
0
100
200
300
400
500
600
700
Time

NYU

Thank You!

~~Q&A Session~~

RNN



Microsoft Stock Price
Real
Predicted
Microsoft Stock Price
Time
8
6
4
2
0
−2
−4
−6
−8
0
100
200
300
400
500
600
700

NYU

Clear and Present Danger: Dataset Bias in Classification Models

Julie E. Cestaro
CSCI-GA 2565 Final Project Presentation
Group Project ID: 20

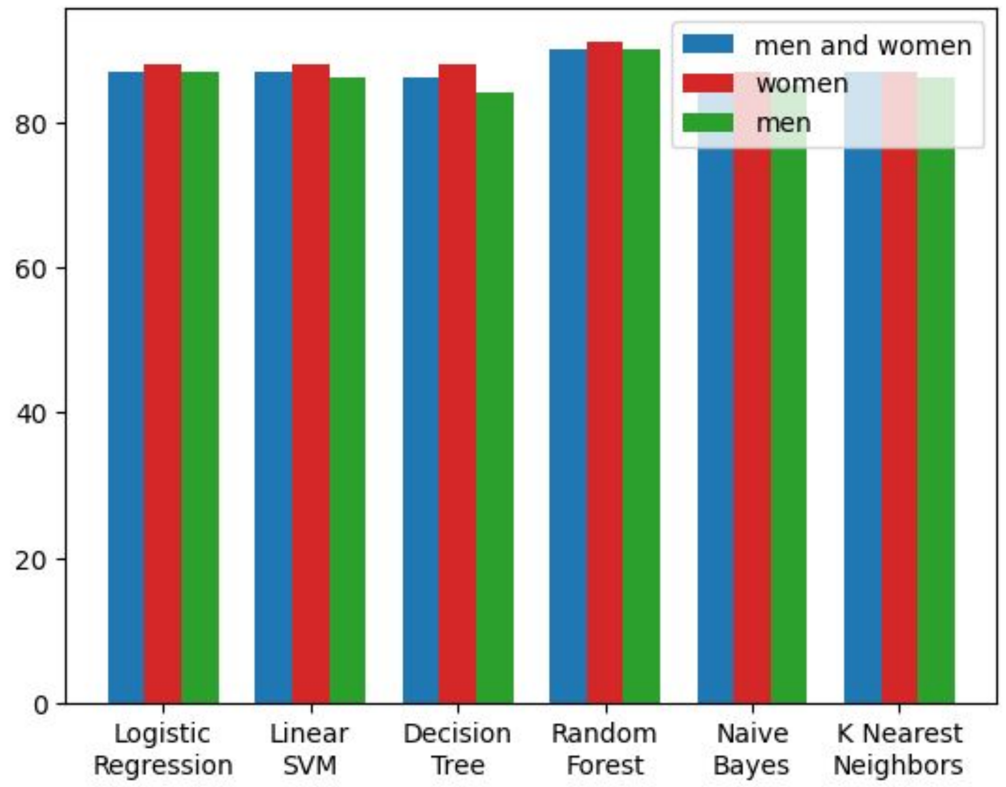
Introduction & Motivation

- Studying ethics and responsible machine learning
- Avoid perpetuating historical biases
- Previous examples
  - Racial bias in criminal justice [1]
  - Color bias in facial recognition [2]
  - Gender bias in word embeddings [3]

1. https://www.liebertpub.com/doi/full/10.1089/big.2016.0047?casa_token=qQna8goMBbsAAAAA%3A-kjbYeNRVLpRXqDHt81Xn2yw0D3YzBzAqRWHYMOVWc9uO1XSRMDKUCOSkVWPJ4OmyGUCMuAUpbyV
2. http://proceedings.mlr.press/v81/buolamwini18a.html
3. https://proceedings.neurips.cc/paper_files/paper/2016/hash/a486cd07e4ac3d270571622f4f316ec5-Abstract.html

Method Overview

- Hypothesis: models trained on biased data directly reflect those biases in their predictions
- Dataset: Adult Census Data from the UCI Machine Learning Repository
    - Income classification
- Set a baseline by training with a dataset manipulated to be evenly split between men making both above and below $50k a year and women making both above and below $50k a year

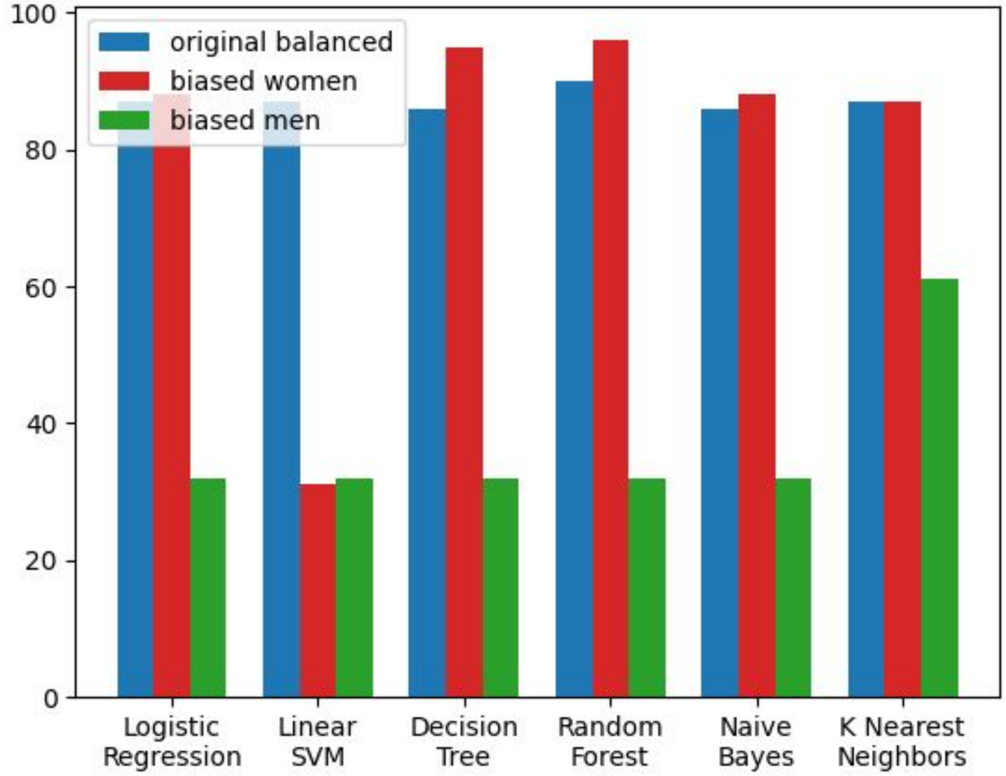


Accuracy of various classifiers trained on the balanced dataset

Method Overview

- Bias the dataset by sampling women making both above and below \$50k a year but only men making below \$50k a year

- Train models using biased dataset and predict on everyone



Accuracy of various classifiers trained on the biased dataset

Conclusions

- Importance of fair and unbiased representation in data
- Models will pretty explicitly reflect any bias you teach it

Thank you! Any questions?

Plant Seedlings Classification

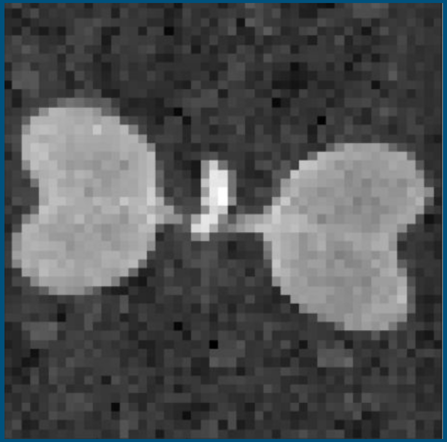
Dean Sheng and Xinhao Liu

Introduction

- 960 distinct plants
- 5539 images
  - 10 pixels/mm
- Several growth stages
- 12 species
  - Sugar beet, Small-flowered Cranesbill, Scentless Mayweed, Shepherd's Purse, Maize, Loose Silky-bent, Fat Hen, Common Chickweed, Common wheat, Charlock, Cleavers, Black-grass
- Which species is the plant?

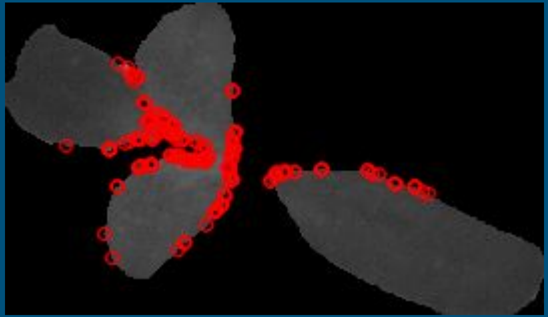
Introduction

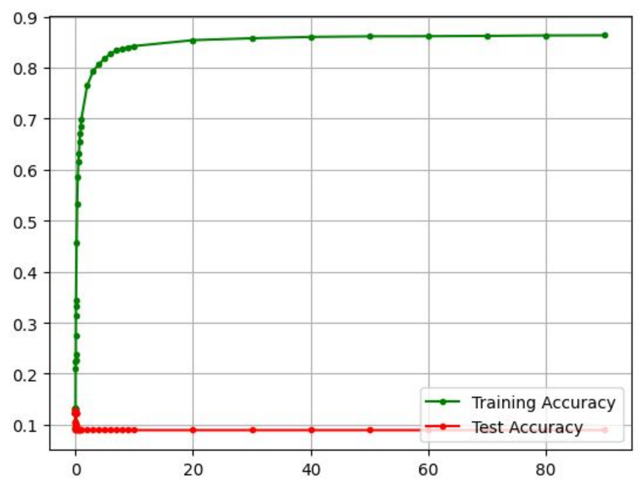
- Observation
  - Plants green
  - Background not green
- Defined greenness: 2G-R-B
- Images with three channels RGB -> Images with a single channel
- Resized images
  - Smallest image
  - 49x49mm
- Normalized greenness value



Another Possible Feature

- Bag-of-words model with ORB descriptors
- Detect keypoints on the edge and corners
- Performed worse than the greeness feature with the same model



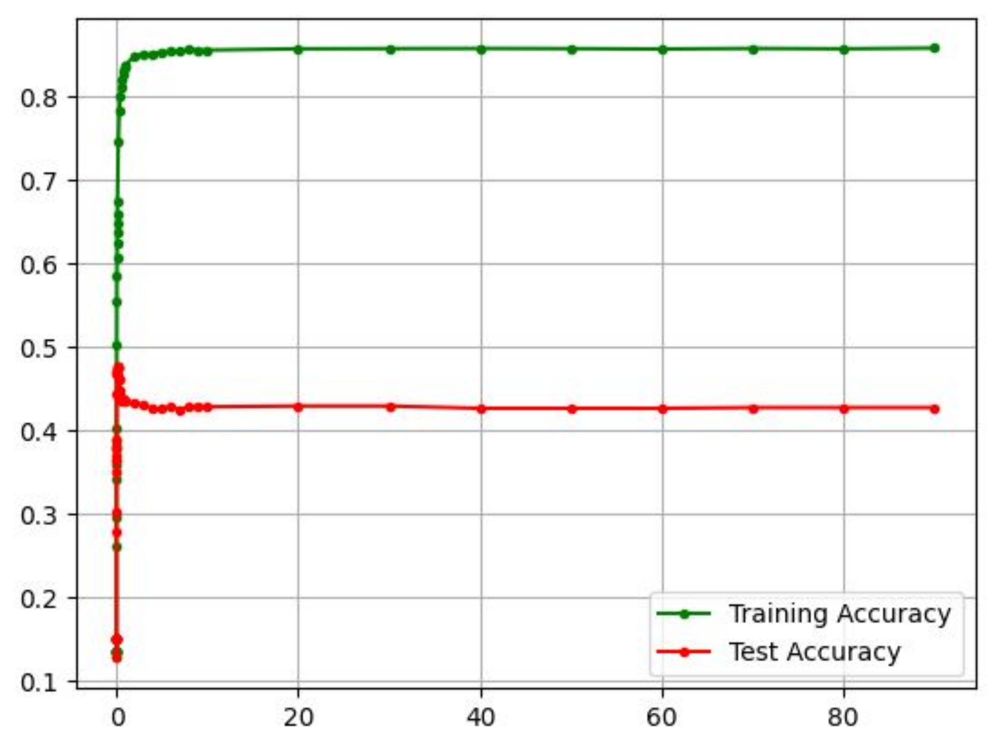


Supervised Analysis & Table of Results

- Training and test dataset
  - Randomly divided
  - Ratio 8:2
- Different
  - Learning models
    - Logistic Regression
    - SVM
    - Neural Networks
  - Feature transformations
  - Regularization techniques

Supervised Analysis & Table of Results

Logistic Regression



Logistic Regression with Lasso Regularization

Supervised Analysis & Table of Results

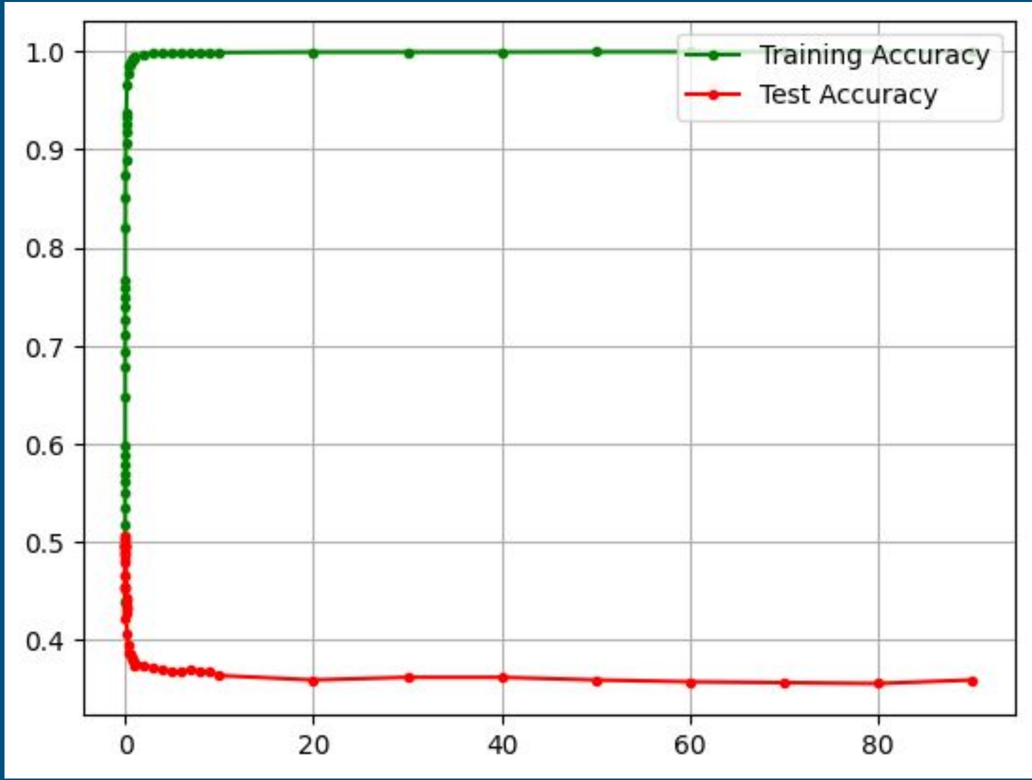
Logistic Regression

The best testing accuracy is 0.4756 using C=0.05
The training accuracy using C=0.05 is 0.606.

C	Training Accuracy	Testing Accuracy
0.0001	0.134507	0.149819
0.0002	0.12909	0.127256
0.0003	0.134507	0.149819
0.0004	0.134507	0.149819
0.0005	0.134507	0.149819
0.0006	0.134507	0.149819
0.0007	0.134507	0.149819
0.0008	0.134507	0.149819
0.0009	0.134507	0.149819
0.001	0.134507	0.149819
0.002	0.134507	0.149819
0.003	0.261792	0.277978
0.004	0.295644	0.302347
0.005	0.341232	0.349278
0.006	0.357933	0.362816
0.007	0.365606	0.369134
0.008	0.378244	0.378159
0.009	0.387949	0.380866
0.01	0.401941	0.388087
0.02	0.501241	0.444043
0.03	0.554954	0.468412
0.04	0.583615	0.467509
0.05	0.605958	0.475632
0.06	0.624013	0.474729
0.07	0.637102	0.475632
0.08	0.647032	0.473827
0.09	0.658542	0.475632
0.1	0.674566	0.473827
0.2	0.745204	0.464801
0.3	0.782216	0.460289
0.4	0.800497	0.447653
0.5	0.810201	0.435018
0.6	0.820131	0.439531
0.7	0.826224	0.435921
0.8	0.82961	0.436823
0.9	0.835252	0.434116
1	0.837283	0.434116
2	0.848116	0.43231
3	0.849695	0.429603

Supervised Analysis & Table of Results

Logistic Regression



Logistic Regression with Ridge Regularization

Supervised Analysis & Table of Results

Logistic Regression

The best testing accuracy is 0.5054 using C=0.004.
The training accuracy using C=0.004 is 0.694.

C	Training Accuracy	Testing Accuracy
0.0001	0.438276	0.422383
0.0002	0.489957	0.454874
0.0003	0.517716	0.464801
0.0004	0.533965	0.478339
0.0005	0.549989	0.487365
0.0006	0.561047	0.493682
0.0007	0.570074	0.49639
0.0008	0.578425	0.495487
0.0009	0.588129	0.495487
0.001	0.598285	0.49639
0.002	0.647935	0.502708
0.003	0.678402	0.505415
0.004	0.693974	0.505415
0.005	0.711578	0.501805
0.006	0.726698	0.497292
0.007	0.739336	0.49639
0.008	0.749041	0.490072
0.009	0.759197	0.487365
0.01	0.767547	0.481047
0.02	0.820131	0.464801
0.03	0.851726	0.454874
0.04	0.874069	0.452166
0.05	0.889867	0.442238
0.06	0.906116	0.440433
0.07	0.918077	0.435921
0.08	0.924848	0.43231
0.09	0.932747	0.430505
0.1	0.936809	0.427798
0.2	0.965471	0.40704
0.3	0.976755	0.394404
0.4	0.984879	0.388087
0.5	0.987136	0.385379
0.6	0.989619	0.385379
0.7	0.99165	0.381769
0.8	0.992327	0.379964
0.9	0.992778	0.374549
1	0.993907	0.377256
2	0.996615	0.374549
3	0.997743	0.370939

Supervised Analysis & Table of Results

Logistic Regression

At best testing accuracy:

- Large difference (>0.1) between training accuracy and testing accuracy
  - Lasso
  - Ridge
- Testing accuracies are not satisfactory (<0.6)

Conclusions:

- Logical regression model is
  - Overfitting
  - Underfitting

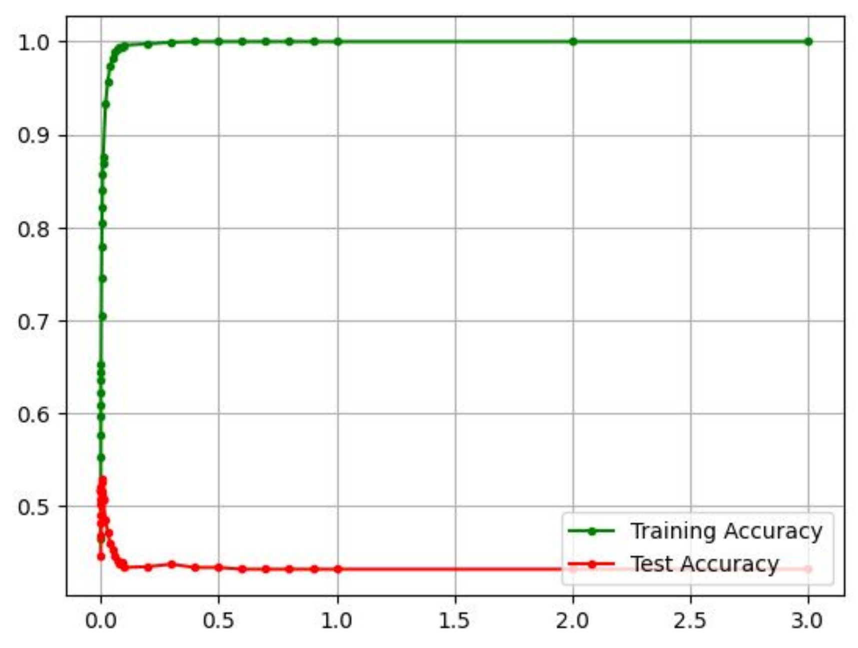
Supervised Analysis & Table of Results

SVM

- Linear
- Polynomial
- Radial basis function kernels
- Ridge Regularization
- Diff values of C
  - Inverse of λ

Supervised Analysis & Table of Results

SVM



Support-vector machines using linear kernel

Supervised Analysis & Table of Results

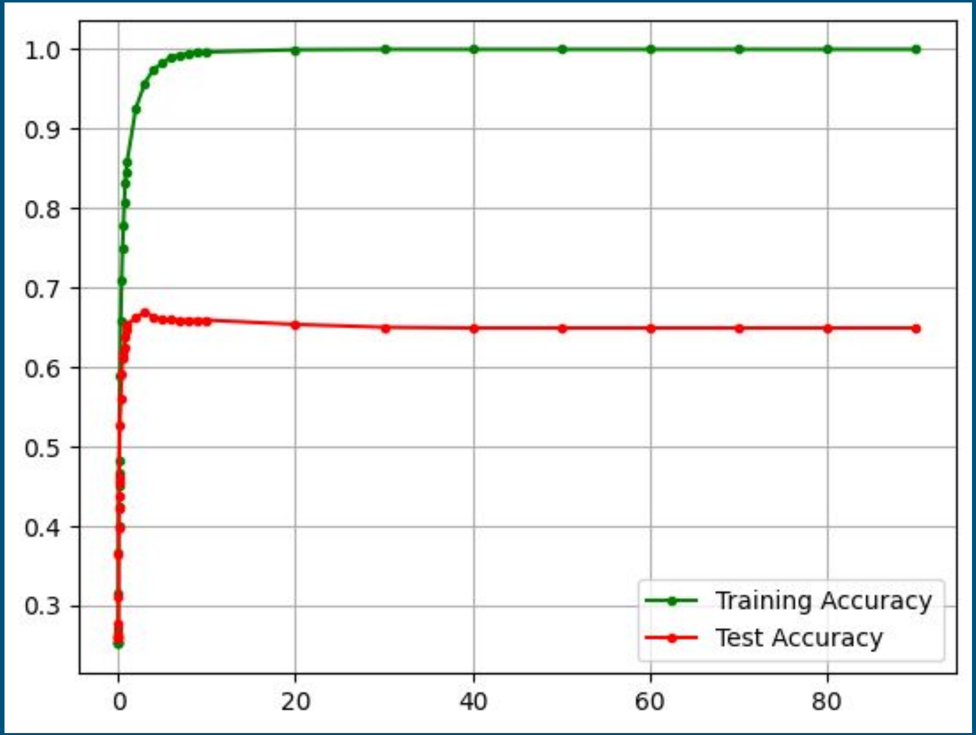
SVM

The best testing accuracy is 0.5289 using C=0.003.
The training accuracy using C=0.002 is 0.7459.

C	Training Accuracy	Testing Accuracy
0.0001	0.464681	0.446751
0.0002	0.517716	0.469314
0.0003	0.552471	0.481949
0.0004	0.577071	0.490072
0.0005	0.596254	0.502708
0.0006	0.609569	0.50722
0.0007	0.621981	0.515343
0.0008	0.635071	0.518051
0.0009	0.645001	0.521661
0.001	0.652223	0.517148
0.002	0.70571	0.526173
0.003	0.745881	0.528881
0.004	0.779057	0.515343
0.005	0.804107	0.508123
0.006	0.822162	0.509025
0.007	0.839765	0.506318
0.008	0.857143	0.512635
0.009	0.868427	0.508123
0.01	0.876552	0.508123
0.02	0.932521	0.484657
0.03	0.957572	0.472022
0.04	0.974498	0.459386
0.05	0.982622	0.453069
0.06	0.988265	0.446751
0.07	0.992101	0.441336
0.08	0.993681	0.437726
0.09	0.994809	0.439531
0.1	0.996163	0.434116
0.2	0.997517	0.435018
0.3	0.999323	0.437726
0.4	1	0.434116
0.5	1	0.434116
0.6	1	0.43231
0.7	1	0.43231
0.8	1	0.43231
0.9	1	0.43231
1	1	0.43231
2	1	0.43231
3	1	0.43231

Supervised Analysis & Table of Results

SVM



Support-vector machines using polynomial kernel

Supervised Analysis & Table of Results

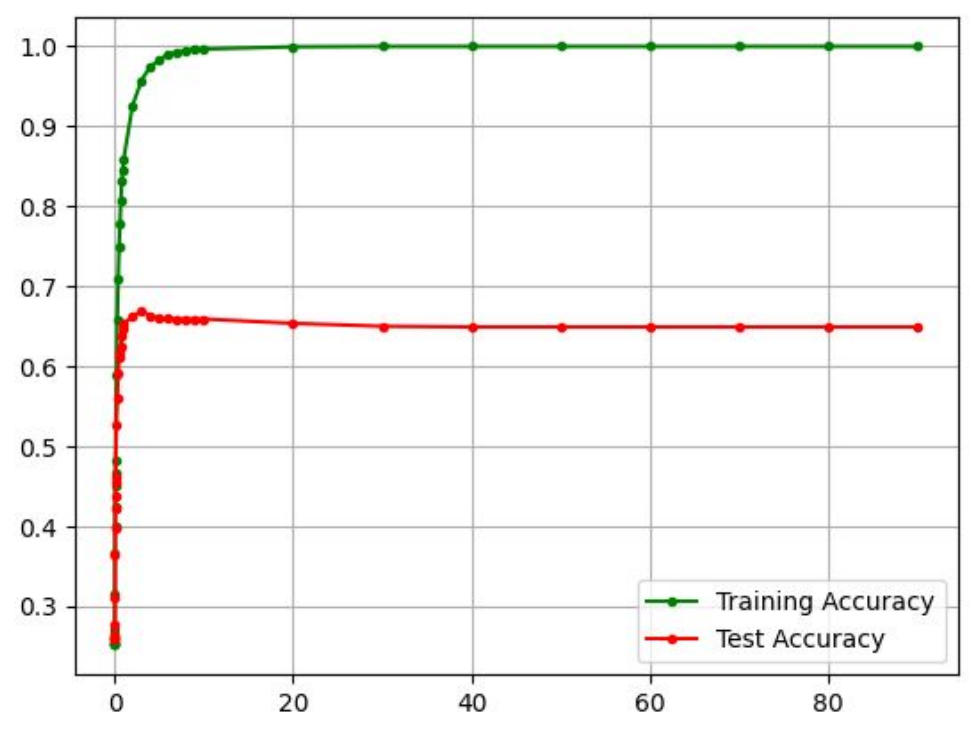
SVM

The best testing accuracy is 0.6083 using C=9.
The training accuracy using C=9 is 0.9352.

C	Training Accuracy	Testing Accuracy
0.007	0.143083	0.150722
0.008	0.146468	0.152527
0.009	0.149176	0.152527
0.01	0.151433	0.154332
0.02	0.189348	0.176895
0.03	0.224554	0.212996
0.04	0.256376	0.232852
0.05	0.288197	0.248195
0.06	0.309637	0.267148
0.07	0.332882	0.278881
0.08	0.35387	0.295126
0.09	0.371474	0.304152
0.1	0.388174	0.318592
0.2	0.491086	0.377256
0.3	0.560144	0.411552
0.4	0.606184	0.437726
0.5	0.643647	0.465704
0.6	0.675243	0.481949
0.7	0.699842	0.495487
0.8	0.722636	0.50722
0.9	0.740465	0.518953
1	0.75694	0.530686
2	0.83796	0.58574
3	0.86617	0.599278
4	0.883322	0.607401
5	0.897314	0.606498
6	0.90747	0.607401
7	0.91898	0.603791
8	0.927556	0.607401
9	0.935229	0.608303
10	0.940871	0.607401
20	0.986685	0.590253
30	0.995035	0.583032
40	0.997743	0.580325
50	0.998195	0.579422
60	0.99842	0.577617
70	0.998872	0.577617
80	0.999097	0.576715
90	0.999097	0.57852

Supervised Analysis & Table of Results

SVM



Support-vector machines using radial-basis function kernel

Supervised Analysis & Table of Results

SVM

The best testing accuracy is 0.6688 using C=3.
The training accuracy using C=3 is 0.9573.

C	Training Accuracy	Testing Accuracy
0.007	0.251185	0.26083
0.008	0.251862	0.259025
0.009	0.251411	0.259928
0.01	0.251636	0.259928
0.02	0.254344	0.262635
0.03	0.269465	0.277076
0.04	0.314376	0.309567
0.05	0.36538	0.362816
0.06	0.399684	0.398014
0.07	0.423381	0.420578
0.08	0.450237	0.436823
0.09	0.465583	0.454874
0.1	0.481155	0.461191
0.2	0.589032	0.525271
0.3	0.657188	0.559567
0.4	0.709546	0.591155
0.5	0.748815	0.610108
0.6	0.778831	0.617329
0.7	0.806816	0.625451
0.8	0.831866	0.637184
0.9	0.844505	0.646209
1	0.857594	0.65343
2	0.925525	0.661552
3	0.957346	0.668773
4	0.974949	0.663357
5	0.982397	0.659747
6	0.990747	0.659747
7	0.992552	0.658845
8	0.995035	0.658845
9	0.996389	0.658845
10	0.996615	0.658845
20	0.999549	0.65343
30	1	0.649819
40	1	0.648917
50	1	0.648917
60	1	0.648917
70	1	0.648917
80	1	0.648917
90	1	0.648917

Supervised Analysis & Table of Results

SVM

At best testing accuracy:

- Large difference (>0.1) between training accuracy and testing accuracy
  - linear, polynomial and RBF
- Testing accuracies are satisfactory (>0.6)

Conclusions:

- Support-vector machines model is not
  - Overfitting
  - Underfitting
- Small bias and large variance

Supervised Analysis & Table of Results

Neural Networks

- DifferTent
  - Activation Functions
    - with/without regularization term
  - Number of Iterations
  - Number of Neurons in the Hidden Layer

Supervised Analysis & Table of Results

Neural Networks

Activation Function	Regularization term	iterations	neurons in each layer	training accuracy	testing accuracy
Sigmoid	without	100	[2401,1200,3]	0.5865	0.5417
Sigmoid	with	100	[2401,1200,3]	0.5865	0.5451
ReLU	with	100	[2401,1200,3]	0.3197	0.2917
tanh	with	100	[2401,1200,3]	0.3197	0.2917
leaky ReLU	with	100	[2401,1200,3]	0.3197	0.2917
Sigmoid	with	50	[2401,1200,3]	0.5885	0.5521
Sigmoid	with	100	[2401,600,3]	0.5885	0.5451

Supervised Analysis & Table of Results

Neural Networks

- Small difference (<0.05) between
    - Training accuracy
    - Testing accuracy
- Best testing accuracies not satisfactory
- Underfitting
- Large bias
- Small variance

Supervised Analysis & Table of Results

Neural Networks

At best testing accuracy:

- Small difference (<0.05) between training accuracy and testing accuracy
  - all neural networks
- Testing accuracies are not satisfactory (<0.6)

Conclusions:

- Logical regression model is not overfitting
- Logical regression model is underfitting
- A large bias and a small variance

Supervised Analysis & Table of Results

Neural Networks

More Conclusions:

- Performance:
  - Models using sigmoid activation function > Models using other activation functions

Decreasing

- Number of iterations
- Neurons in the hidden layer
- No significant effect on testing accuracy

Supervised Analysis & Table of Results

Neural Networks

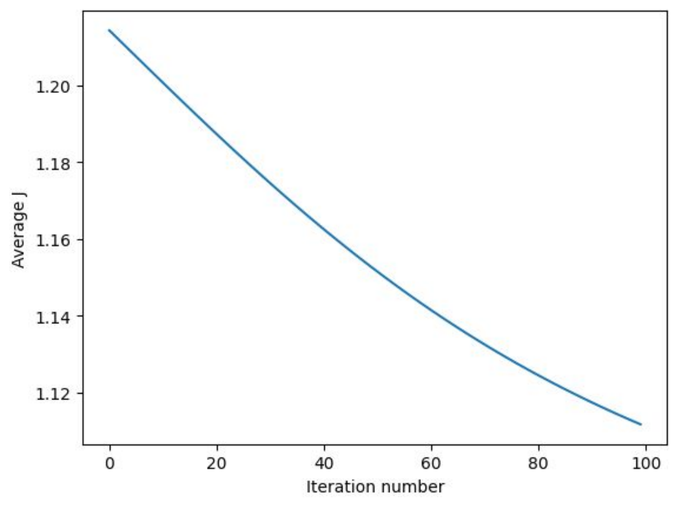
More Conclusions:

- Sigmoid performs the best
- Iterations or neurons in the hidden layer not affect the accuracy

Supervised Analysis & Table of Results

Neural Networks

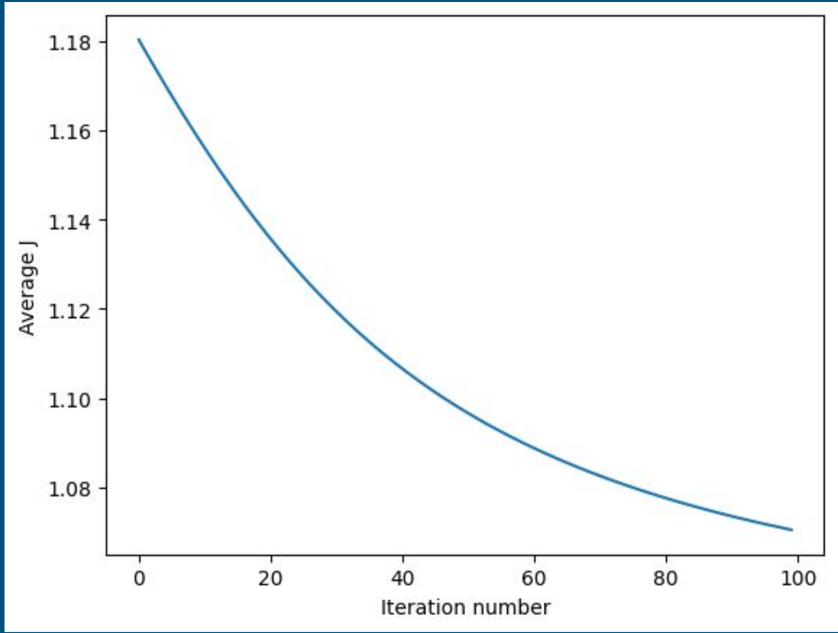
Here are the graphs of how error changed with respect to iterations:



Sigmoid activation function, without regularization term, 100 iterations, 1200 neurons in the hidden layer

Supervised Analysis & Table of Results

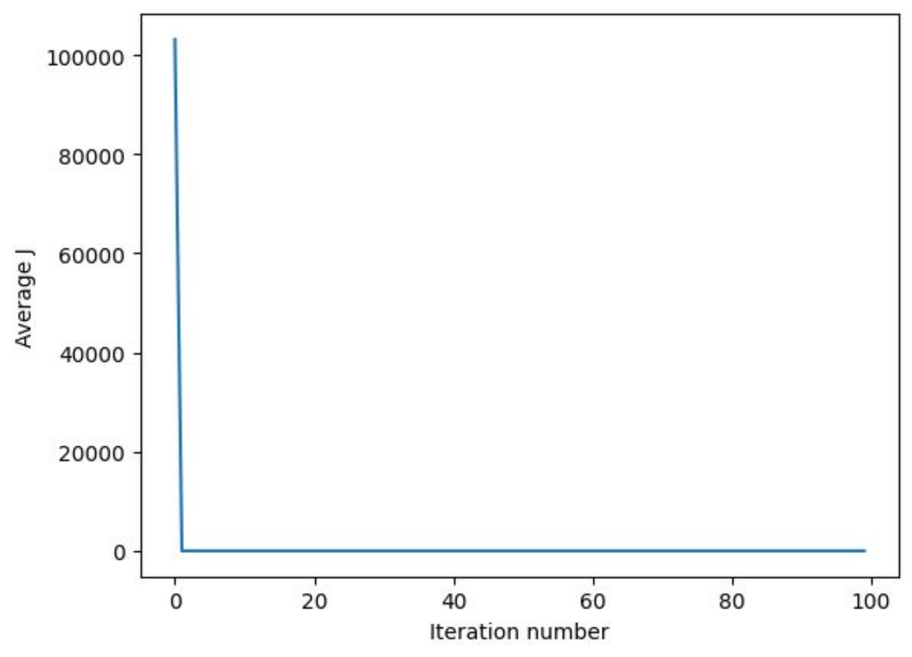
Neural Networks



Sigmoid activation function, with regularization term, 100 iterations, 1200 neurons in the hidden layer

Supervised Analysis & Table of Results

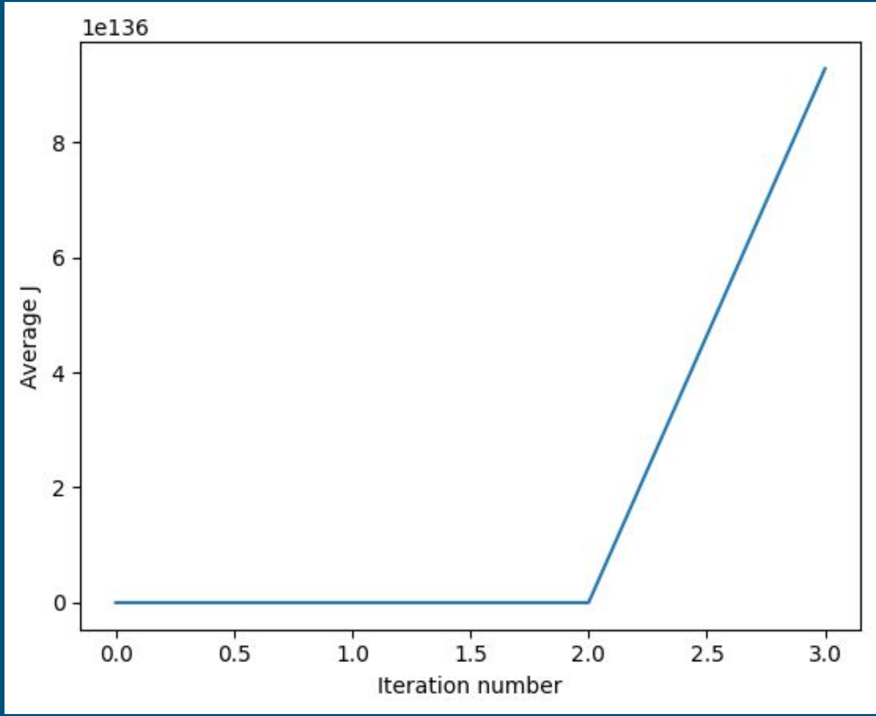
Neural Networks



ReLU activation function, with regularization term, 100 iterations, 1200 neurons in the hidden layer

Supervised Analysis & Table of Results

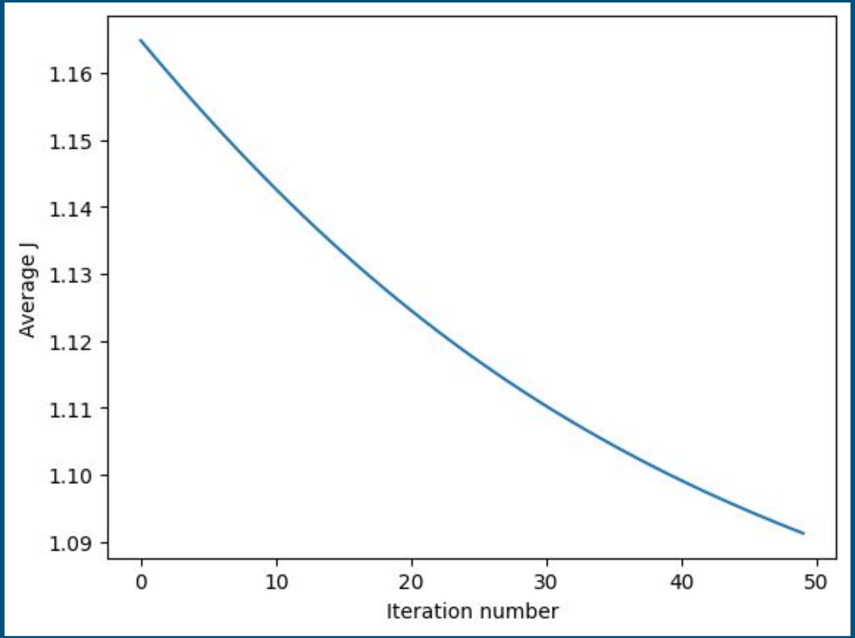
Neural Networks



Leaky ReLU activation function, with regularization term, 100 iterations, 1200 neurons

Supervised Analysis & Table of Results

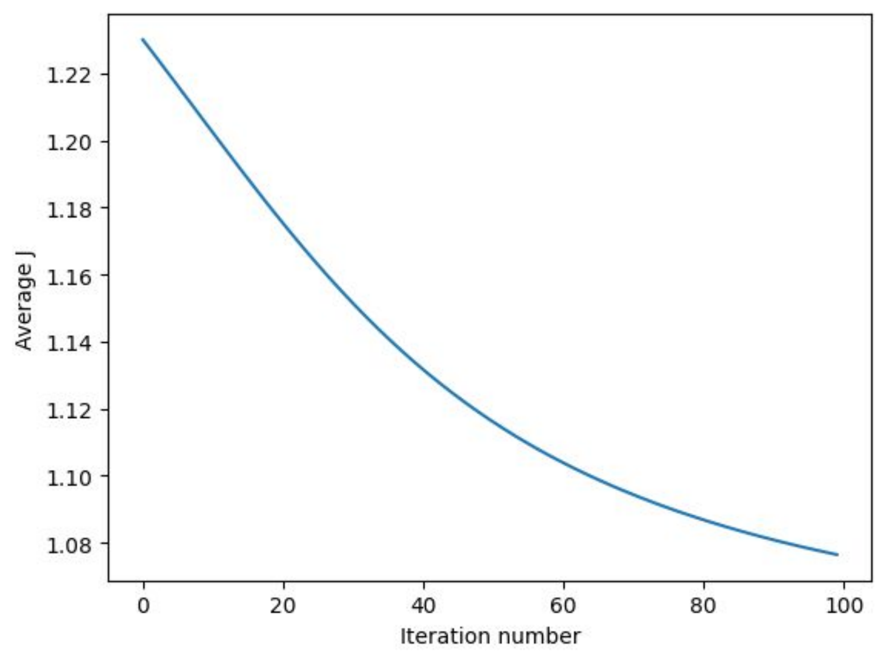
Neural Networks



Sigmoid activation function, with regularization term, 50 iterations, 1200 neurons in the hidden layer

Supervised Analysis & Table of Results

Neural Networks



Sigmoid activation function, with regularization term, 100 iterations, 600 neurons in the hidden layer

Malware Detection using partial polynomial kernel

Jiajie Zou

Background

- Ransomware attacks in Europe saw a 234% increase in 2021, with 68% of organizations in India experiencing at least one ransomware attack.
- The average cost of a ransomware attack on organizations was $1.85 million in 2021. The global cost of cybercrime, including malware, was estimated to be between $1.5 trillion and $2 trillion in 2020. In the United States, ransomware caused an estimated $159.4 billion in downtime in 2021.
- What's needed?
  - Application that can distinguish malware at high accuracy and speed.

Dataset

- Malware: 3565
- Goodware: 899
- Features: 242 features
- Good accuracy without any process using various models
- Accuracy:
  - Logistic Regression: 96.5%
  - Decision Tree: 98.99%
  - Random Forest: 99.44%
  - SVM: 99.55%

Problems with dataset

- Uneven distribution between malware and goodware.
  - Malware roughly 4 times as goodware
  - Different distribution of malware and goodware in reality
    - Most softwares are good, only a small portion of the softwares is bad
    - If goodware:malware is 100:1, the malware detection application detects malware with 100% accuracy and mislabel goodware as malware at 1% error rate, half of the reported malwares are actually good
  - Avoid misclassifying goodware as malware as much as possible
  - 5000 times more penalty on false positives than false false negatives
- More than 200 features:
  - Very hard to get so many features for malware detection in reality.

Feature selection using RFE

- Recursive Feature Elimination: a feature selection method to identify a dataset's key features.
- Problems:
    - Different feature selected for different model
    - The intersection of the sets each containing 40 most important feature contains only 14 feature
    - The features selected model based, doesn’t intrinsically represent the best features to distinguish malwares from goodwares.
- Solutions: Using feature selection techniques that’s not model based, fit the models on the selected features, and see the accuracy of each model.

Feature selection using Lasso

- More robust, not model based
- By Controlling the alpha value, we can adjust the number of selected feature
    - Alphas:
        - 0.001,0.002,0.005,0.01,0.02,0.1,0.11
    - The number of selected features:
        - 44,33,23,10,5,2,1
- Good accuracies for various models when selected feature are more than 10.
- Observations:
    - Better accuracies using Decision Tree and Randomforest
        - Randomforest and decision tree are not a stable algorithm intrinsically
    - Worse accuracies using SVM and logistic regression
        - SVM and logistic regression are more stable algorithms
    - Implication: XOR situation
    - **Solution: construct polynomial kernels using features**

Problems with polynomial construction

- Polynomial of degree d
- The dimension of kernel grows roughly 2^d if the number of features used for polynomial kernel construction doubles.
- Typical kernel is of degree 2 to 4. Double the size of features for kernel construction leads to 4 to 16 times more work.
- Potential solution:
  - Gaussian kernel:
    - **Doesn’t work well**
  - Take out some features that are more “important” to construct the polynomial kernel, and concatenate the polynomial kernel with the remaining features.

Gaussian kernel result

- Features 44
  - SVM Accuracy: 97.98%
- Features 33
  - SVM Accuracy: 95.74%
- Features 23
  - SVM Accuracy: 94.84%
- Features 10
  - SVM Accuracy: 78.95%

Experiments

- Total features: [44,33,23,10]
- Polynomial features: [44,33,23,10]
- Degrees of polynomial: [2,3,4]

Result of this approach 1

- Total features: 33
- Degree: 3
- Polynomial features: 33
  - Accuracy:
    - Logistic regression: 99.55%
    - SVM: 99.21%
- Polynomial features: 23
  - Accuracy:
    - Logistic regression: 99.21%
    - SVM: 99.32%
- Polynomial features: 10
  - Accuracy:
    - Logistic regression: 97.53%
    - SVM: 98.65% (**beat gaussian kernel with 44 features)**
- Baseline using 33 features without kernel:
  - Logistic regression: 94.06%
  - SVM: 95.74%

Result of this approach 2

- Total features: 23
- Degree: 3
- Polynomial features: 23
    - Accuracy:
        - Logistic regression: 99.21%
        - SVM: 99.21%
- Polynomial features: 10
    - Accuracy:
        - Logistic regression: 97.42%
        - SVM: 98.21%

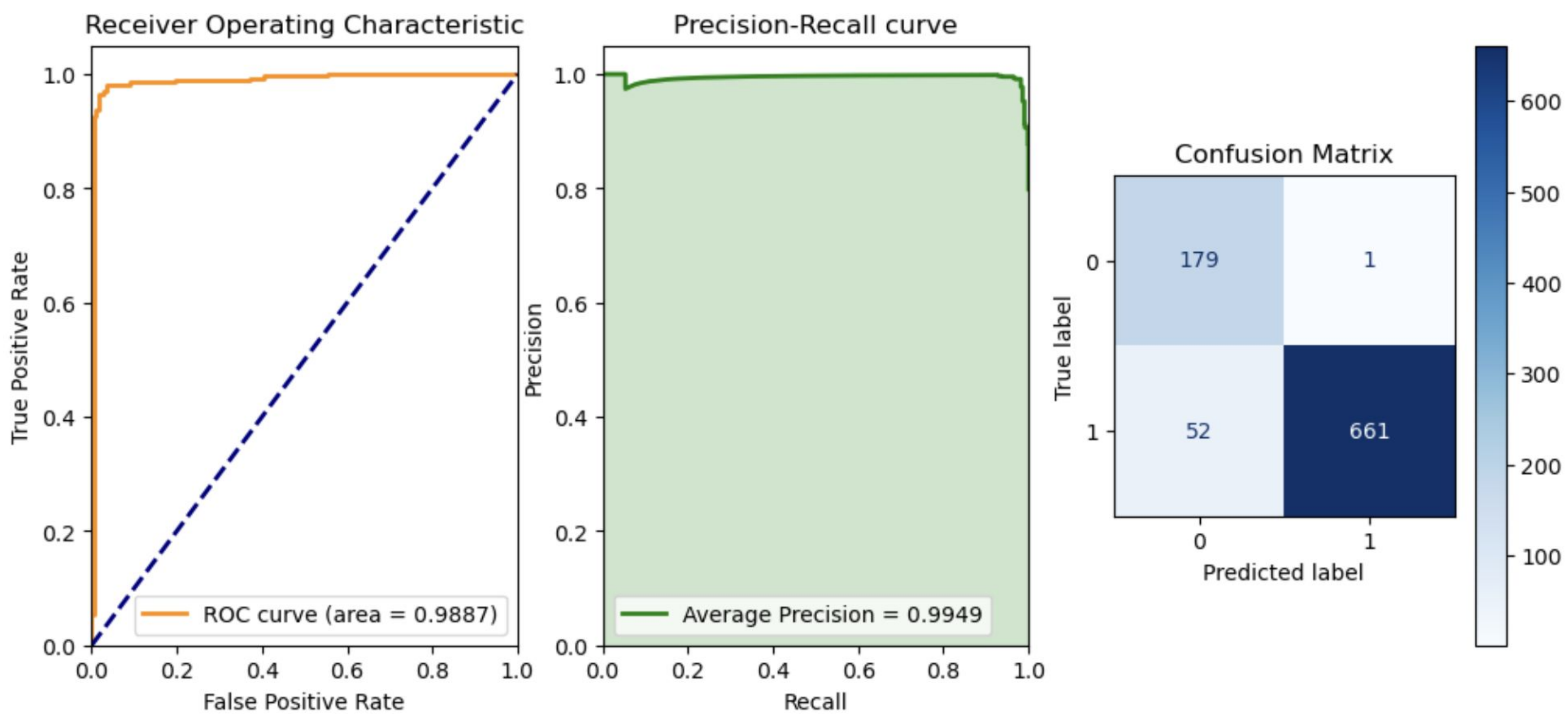
Result of this approach 3

- Scenario 1:
  - Total features: 10
  - Polynomial features: 10
  - Degree 3:
    - Accuracy:
      - Logistic regression: 84.65%
      - SVM: 85.33%
- Scenario 2 (without polynomial kernel):
  - Features: 23
  - Accuracy:
    - Logistic regression: 94.28%
    - SVM: 94.84%

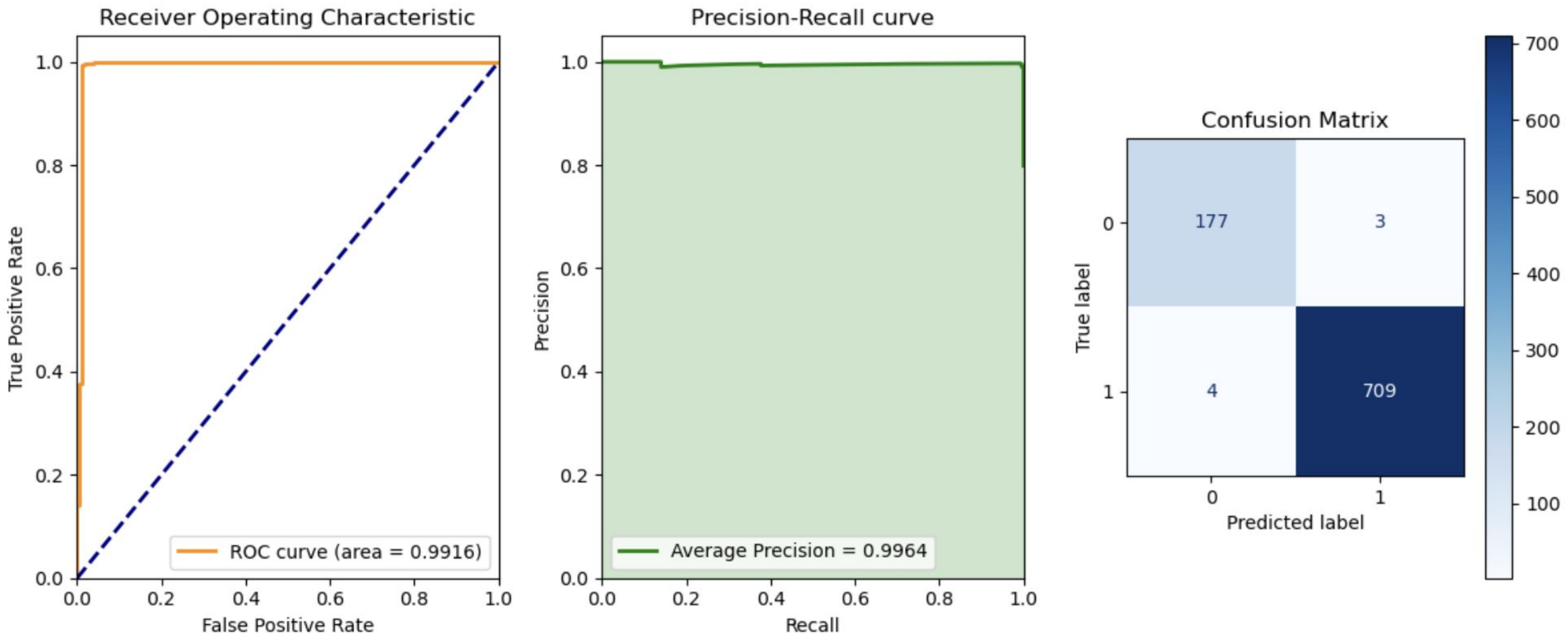
Analysis

- We see that when keep the all features at 33, change the features to construct polynomial kernel from 33 to 23 doesn’t degenerate the accuracy much, it even improves the accuracy of the svm.
  - This approach is comparable to the baseline, which uses more than 200 features to train the model
- Total feature 23, polynomial feature 10 has a
  - better accuracy than:
    - Only the polynomial kernel with 10 features
    - Only the 23 features without kernel
  - One eighth of the time compared to using all 23 features to construct kernel

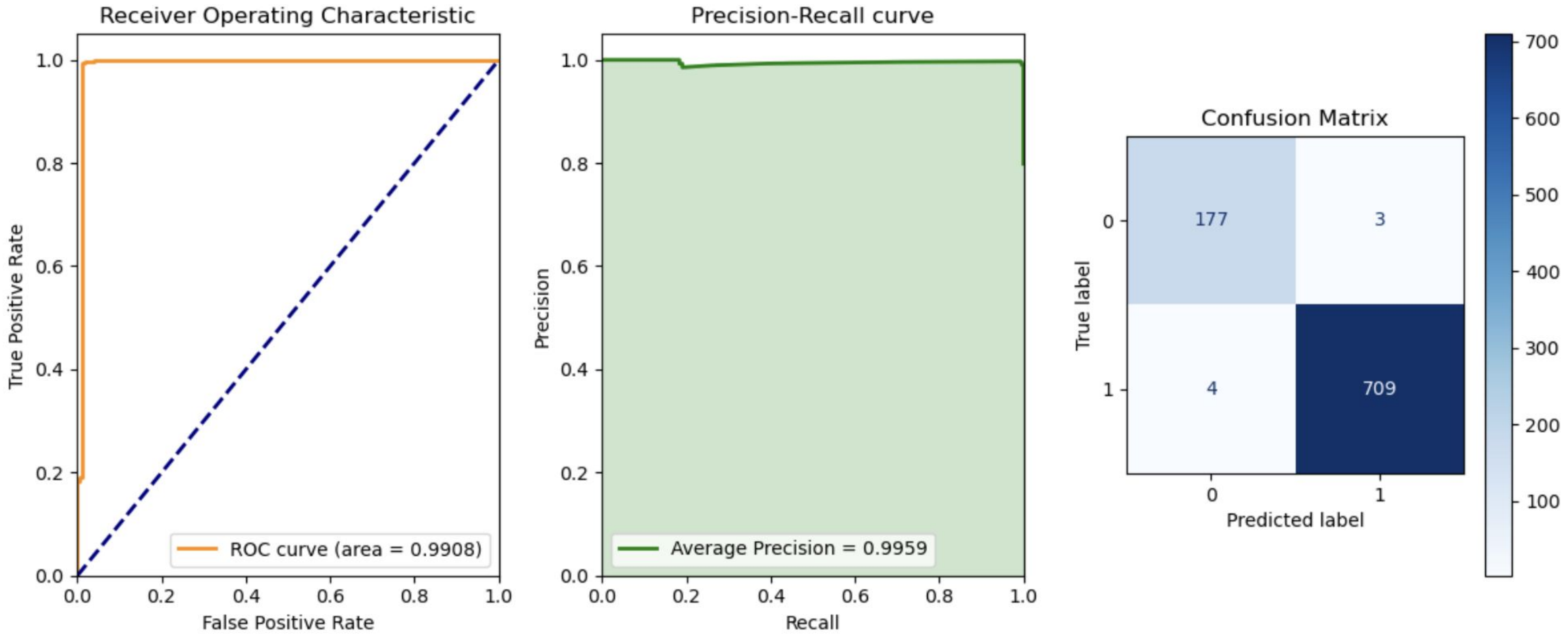
Total feature 33 without polynomial kernel



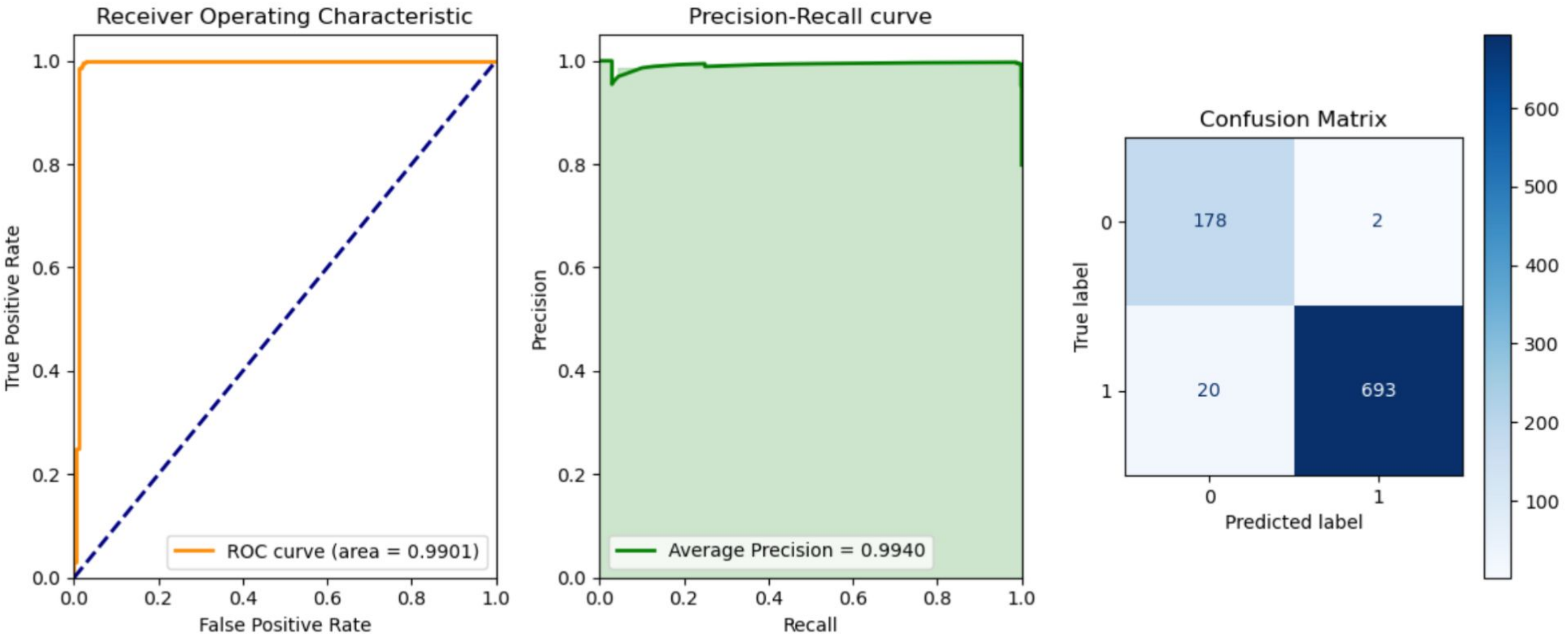
Total feature 33 with polynomial feature 23



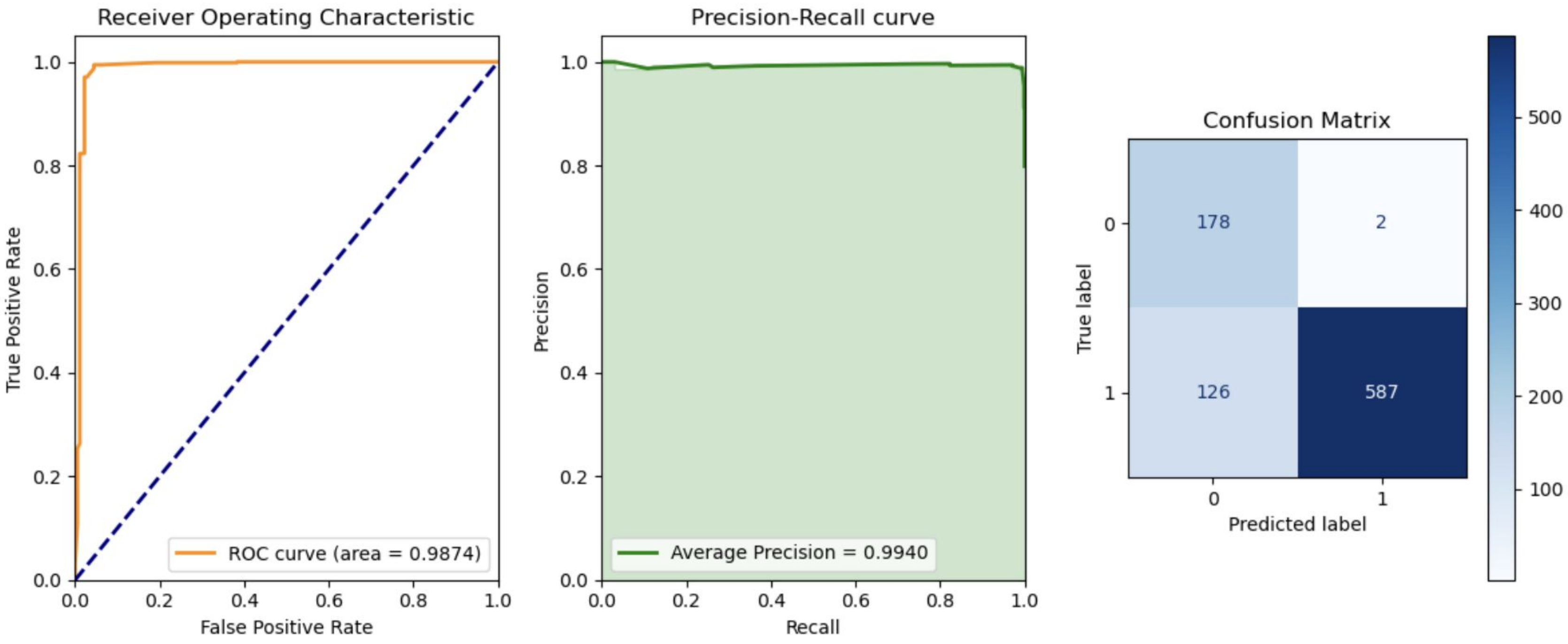
Total feature 23 with polynomial feature 23



Total feature 23 with polynomial feature 10



Total feature 10 with polynomial feature 10



Predicting Patent Quality from Text Data

2023F Machine Learning

Project ID 23: Kyuhun Lee

Background

- Patent data is frequently used in economics and management research as a proxy for technological innovation and knowledge stock
- Patents vary in the scope of their claims, the technological domain, and the importance and quality of the invention they protect
- In other words, some patents are better than others
- How can we measure the quality or value of a patent?

Citations as a Measure of Patent Quality

- Forward citation counts (e.g., Hall, Jaffe, & Trajtenberg, 2005)
  - Consistently recorded by USPTO only after 1947
  - Discrete values: can cause problems for low-citation inventions
  - Relies on the discretion of the inventor or the examiner
  - Typically needs >5 years for citations to accumulate, thus uninformative for recent patents

Idea

- Would a ML model be able to predict patent quality based on textual data contained in the focal patent?
- How much information would text data add to known predictors of quality?

Data and Model

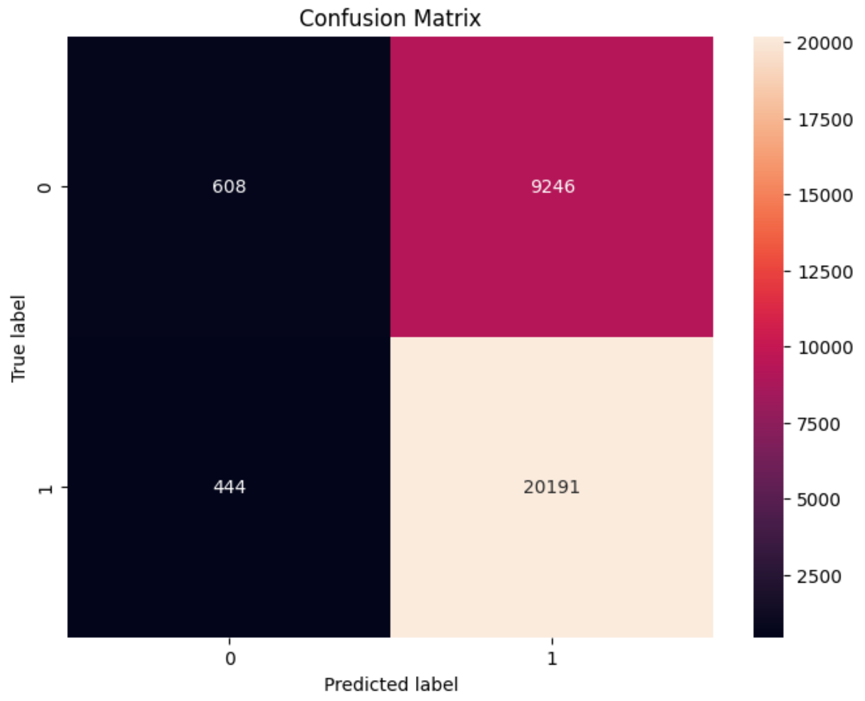
- Input Data
  - ≈100,000 (out of approx. 1M) patent documents in CPC Class G06 (Computing; Calculating or Counting)
  - Obtained word embeddings from Logic Mill (BERT-based model pre-trained for patents, etc.)
  - Combined with year dummies, number of claims, and number of backward citations
- Output Data
  - Citations received in 5 years since grant date
  - Arcsinh(#cites) for regression
  - 1[#cites>0] for classification

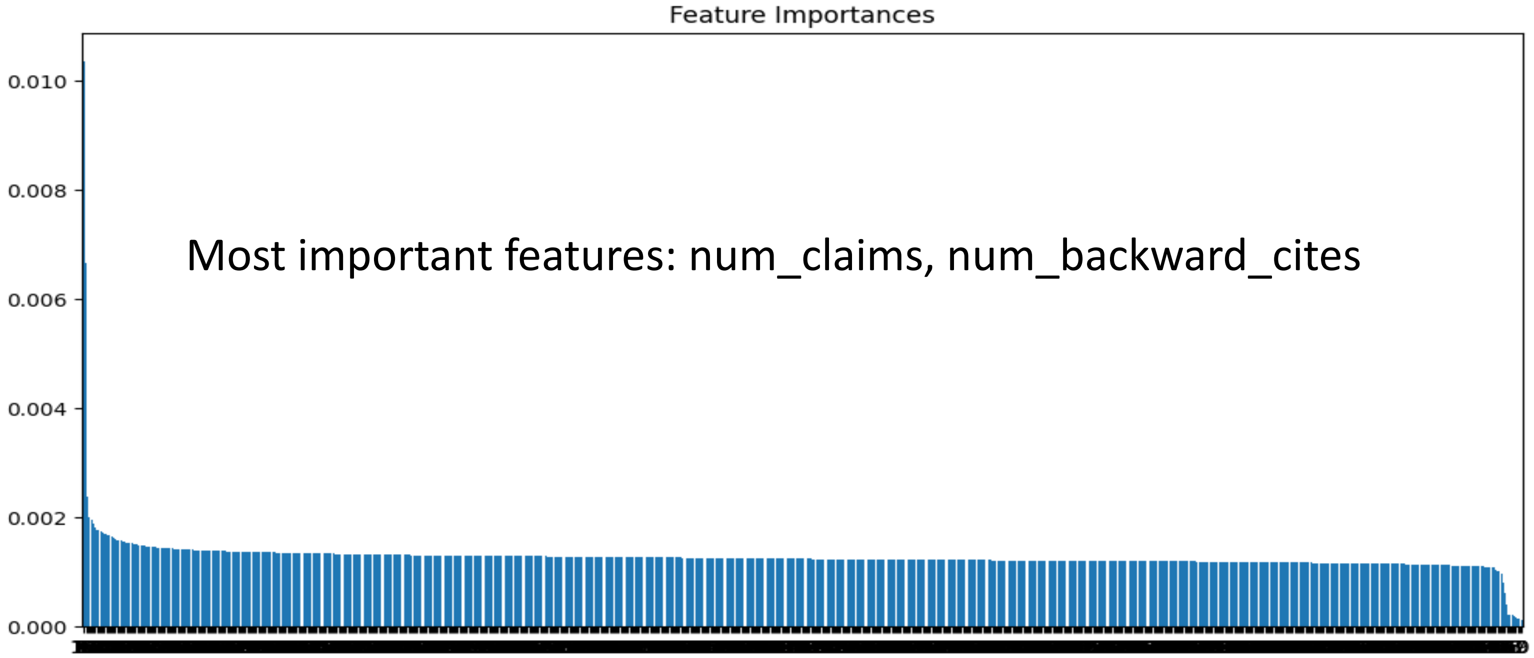
Data and Model

- Model presented today
  - Random Forest Classifier (sklearn.ensemble.RandomForestClassifier)
  - n_estimators = 100
  - min_samples_split = 2

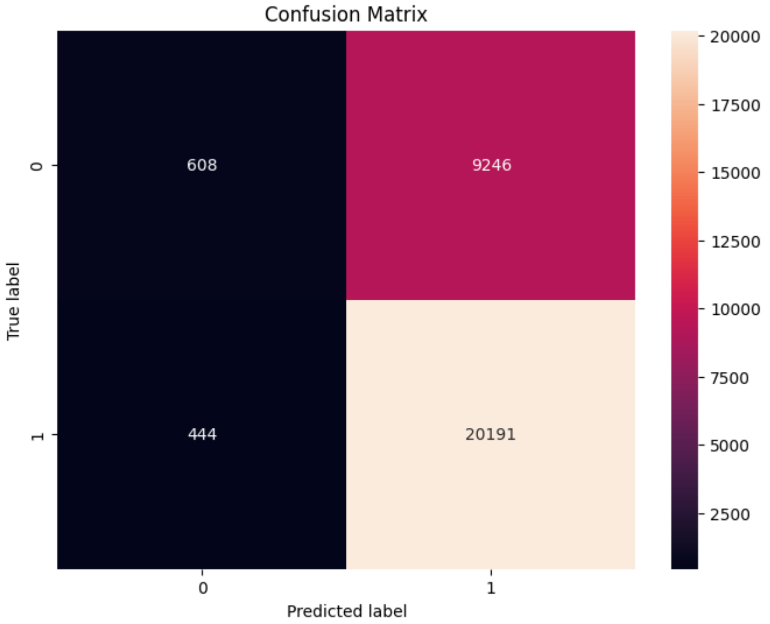
Results

```
Accuracy: 0.6821804585260258
              precision    recall  f1-score   support

           0       0.58      0.06      0.11      9854
           1       0.69      0.98      0.81     20635

    accuracy                           0.68     30489
   macro avg       0.63      0.52      0.46     30489
weighted avg       0.65      0.68      0.58     30489
```





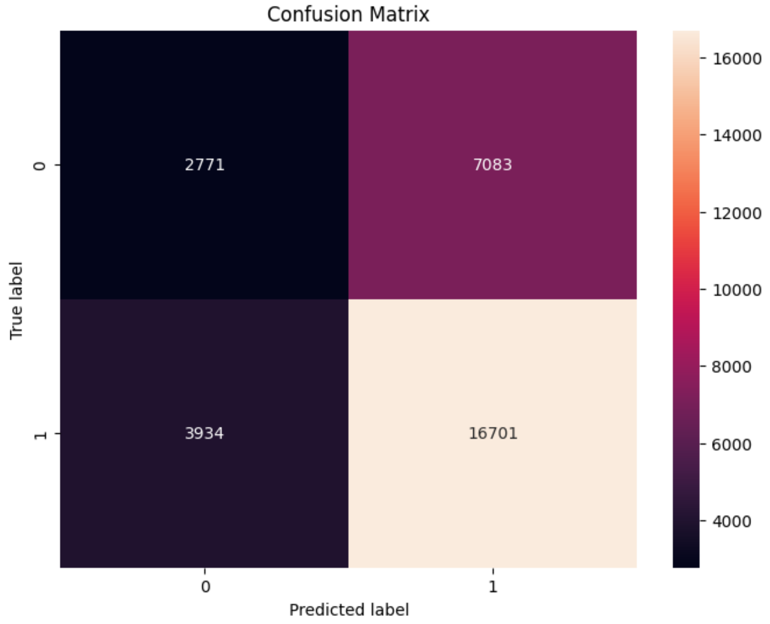
Most important features: num_claims, num_backward_cites



Accuracy: 0.6821804585260258

	precision	recall	f1-score	support
0	0.58	0.06	0.11	9854
1	0.69	0.98	0.81	20635
accuracy			0.68	30489
macro avg	0.63	0.52	0.46	30489
weighted avg	0.65	0.68	0.58	30489

Random Forest with embeddings



Accuracy: 0.6386565646626652

	precision	recall	f1-score	support
0	0.41	0.28	0.33	9854
1	0.70	0.81	0.75	20635
accuracy			0.64	30489
macro avg	0.56	0.55	0.54	30489
weighted avg	0.61	0.64	0.62	30489

Random Forest without embeddings
(year, num_claims, num_backward_cites)

Discussion

- Not a very good model: only slightly better than chance
- Word embeddings don’t seem to carry much information about patent quality (at least on their own)
- Text similarity with previous/future patents may be more informative

The performance of capturing periodicity in User Sequences for Recommendation Click-through Rate Prediction Problems

Project 24

Xin Peng(xp2083@nyu.edu)

23.12.11

PART 01

Introduction

User sparsity problems in Click-through rate problems in recommendation area

Recommendation systems

a subclass of information filtering system that provides suggestions for items that are most pertinent to a particular user

Click-through rate problems

CTR prediction problems is about to predict the probability that users will click on an item

models

Before: Collaborative Filtering, Markov Chain
Currently: ML and DL models
item based features + user based features
user based features: user specific features + user group features

User sparsity problem

The user behavior sequence length was normally short.
In most recommendation datasets, user sequence is on average less than 100[1,2].

NYU

PART 02

Dataset Inspection

There exists periodicity patterns

Periodicity patterns in items

99.98% users have clicked the same item >= 2 times.
99.47% users have clicked the same item >= 3 times.
On average, people click the same item for 1.35 times.
On average, people click the same item in 15 days

summary	count
count	8723505
mean	1.3534585009121907
stddev	1.3953475221169687
min	1
max	173

summary	time_gap
count	1489844
mean	15.523516556095807
stddev	24.31892748209997
min	1
max	182

Periodicity patterns in categories

100% users have clicked the same category >= 2 times.
99.99% users have clicked the same category >= 3 times. On average, people click the same category for 4.29 times.
On average, people click the same category in 19 days.

summary	count
count	1173026
mean	4.299311353712535
stddev	8.645744685128538
min	1
max	182

summary	time_gap
count	567847
mean	19.089863995055005
stddev	31.90577401511904
min	0
max	182

NYU

PART 03

Method

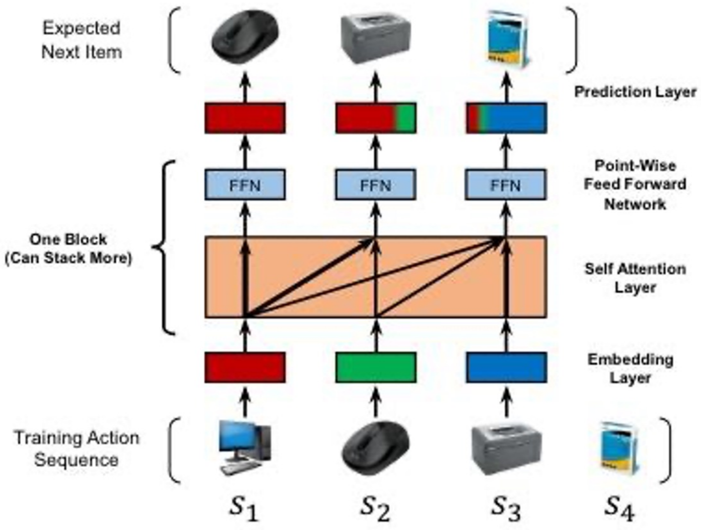
Main Process

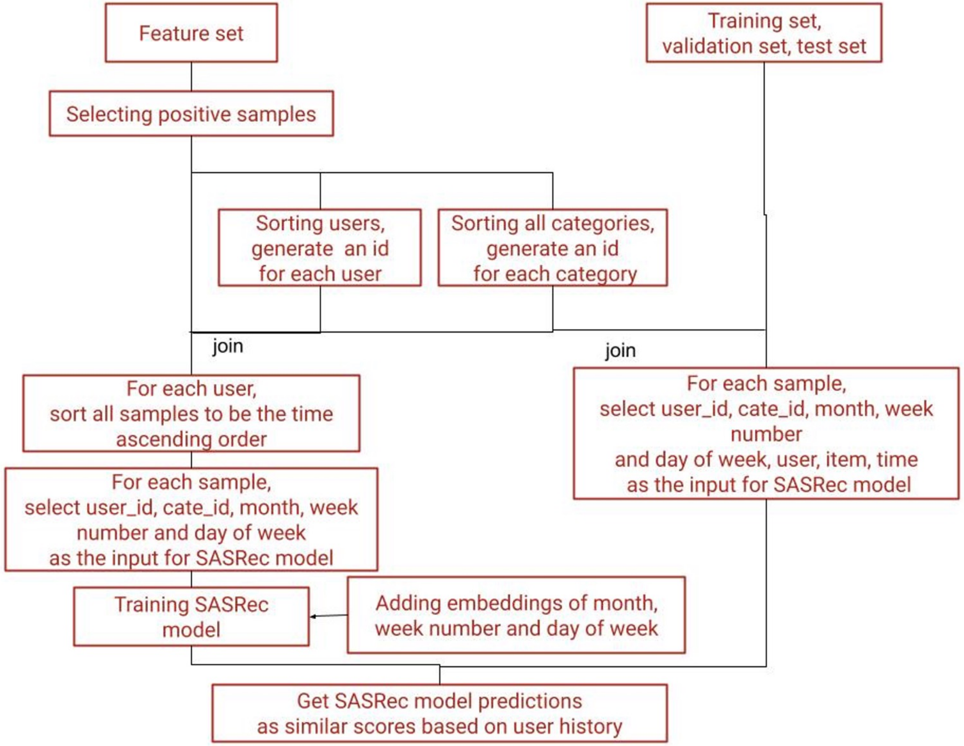
1. Generate features capturing periodicity in user sequences
   1.1 Capture periodicity patterns using models
   1.2 Capture periodicity patterns using statistical methods

1. Build base features

1. Compare model results with or without features capturing periodicity in user sequences

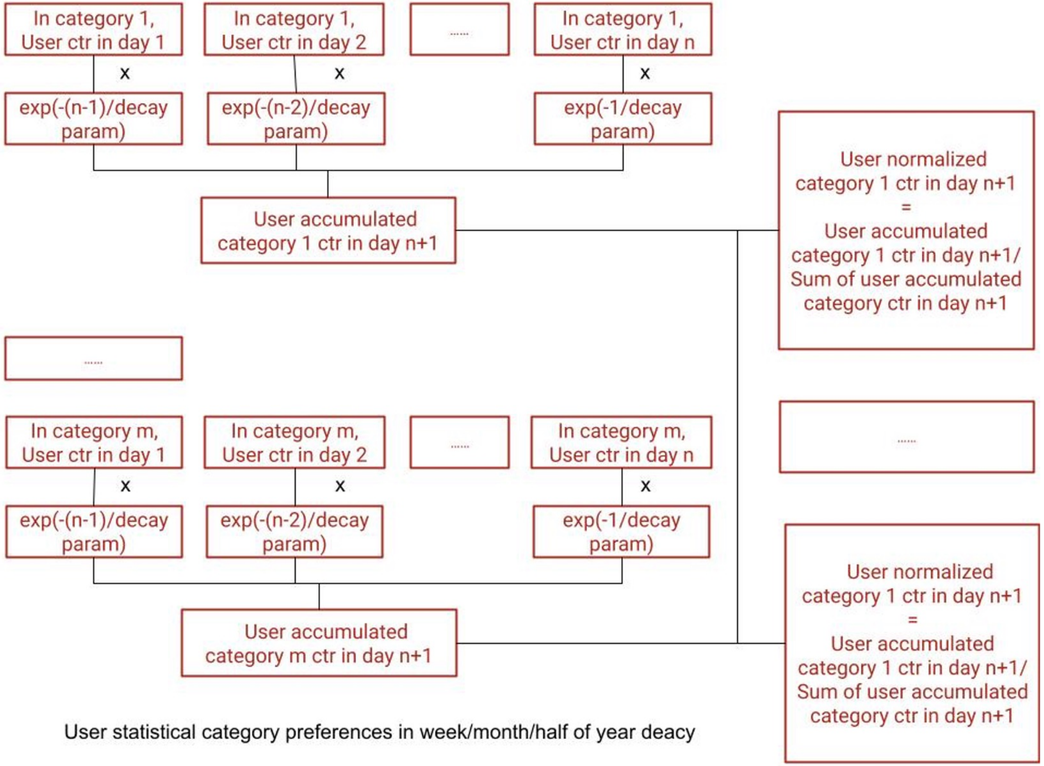
Generate features capturing periodicity in user sequences
By using models





- Add period related embeddings: month, week number, day of week.
- Change granularity from items to categories.

Generate features capturing periodicity in user sequences
By using statistical methods



User statistical category preferences in week/month/half of year deacy

auto-regression (linear combination) and exponential smoothing (exponential decay) for each user's ctr in each category in each day

for each user and each category, accumulate the ctr values in 167 days and normalize it.

decay rate: a week, a month, 6 months

Base features

- item based feature

item ctr feature, category ctr feature, brand ctr feature, seller ctr feature.

- user specific feature

no user group features

user specific features: user ctr feature, user item ctr feature, user brand ctr feature, user seller feature, user hour feature, user weekday feature.

PART 04

Result

Capturing periodicity in user sequences do help

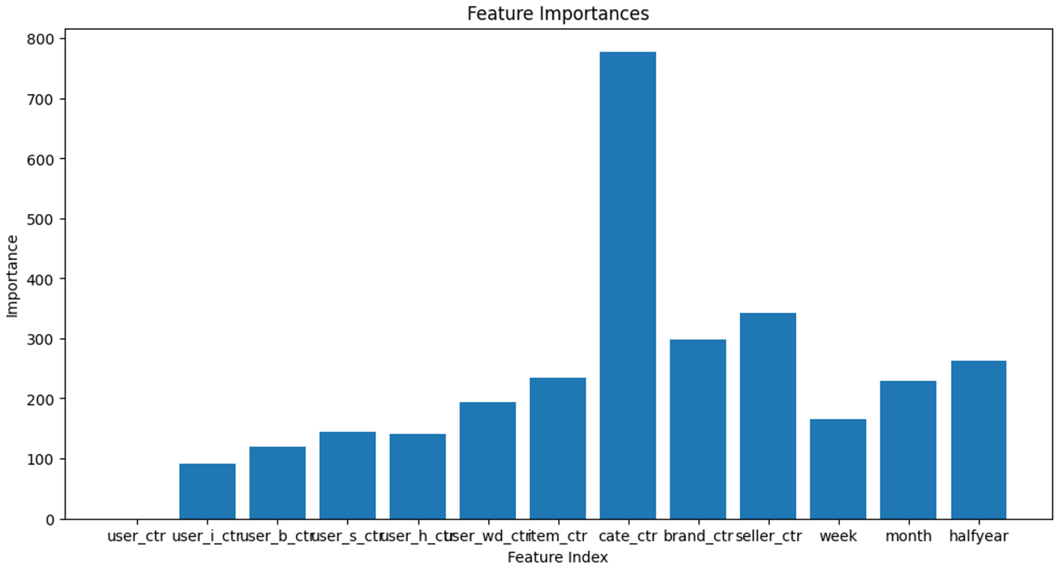
feature	logistic regress test set auc	gbdt test set auc
base features	0.8569	0.9406
base features with SASRec score not capturing periodicity	0.8628	0.9466
base features with SASRec score capturing periodicity	0.8641	0.9484
base features with fixed periods statistical features	0.8720	0.9529

For long sequence users

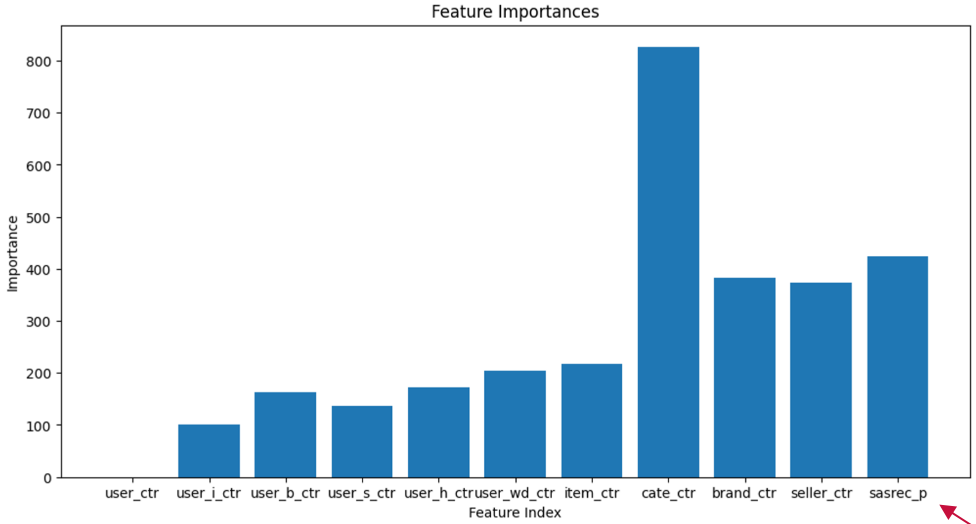
(1) adding user historical sequence features are helpful for improving the ctr prediction results.

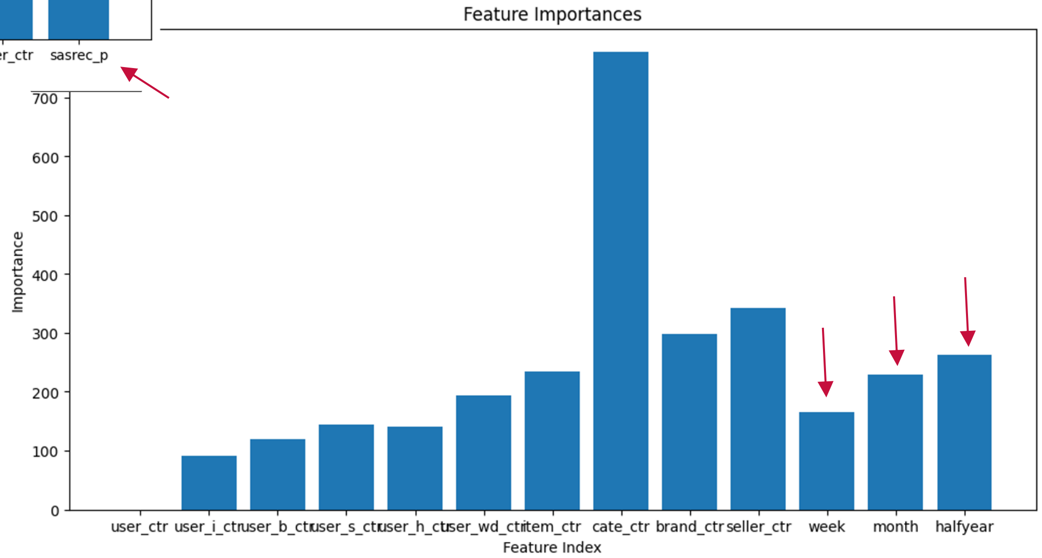
(1) for the SASRec model, capturing periodicity information is better than not capturing periodicity information.

(1) fixed period statistical methods, have a better result than SASRec model in improving the ctr prediction result.



NYU


Feature Importances
Importance
800
700
600
500
400
300
200
100
0
user_ctr
user_i_ctr
user_b_ctr
user_s_ctr
user_h_ctr
user_wd_ctr
item_ctr
cate_ctr
brand_ctr
seller_ctr
sasrec_p
Feature Index


Feature Importances
Importance
700
600
500
400
300
200
100
0
user_ctr
user_i_ctr
user_b_ctr
user_s_ctr
user_h_ctr
user_wd_ctr
item_ctr
cate_ctr
brand_ctr
seller_ctr
week
month
halfyear
Feature Index

NYU

PART 05

Future work

For normal sequence length users

Prove that for normal sequence length users instead of long sequence users,

the features capturing periodicity patterns of user sequences still improve the click-through rate predictions

NYU

Thank you.

NYU

Reference.

[1]
R. He; J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. proceedings of the 25th international conference on world wide web, 2016.

[2]
J. McAuley; C. Targett; Q. Shi; A. van den Hengel. Image-based recommendations on styles and substitutes. Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, 2015.

NYU

Machine Unlearning

Simon Zeng, Project ID #27

Problem: High Impact, No Backtracking

- Machine learning is now used for prediction in a wide range of applications that can lead to significant impacts (ex. Healthcare, education, spending analytics, etc).

- **Potential Problems:**
  - **Privacy Laws.** Individuals have rights to the data encompassing information of themselves/"right to be forgotten" (CCPA, GDPR). This data is used to train many models.
  - **Irrelevant/Outdated Data.** Data may become irrelevant over time.
  - **Sensitive Data.** Passwords, personal identifiable information (PII)

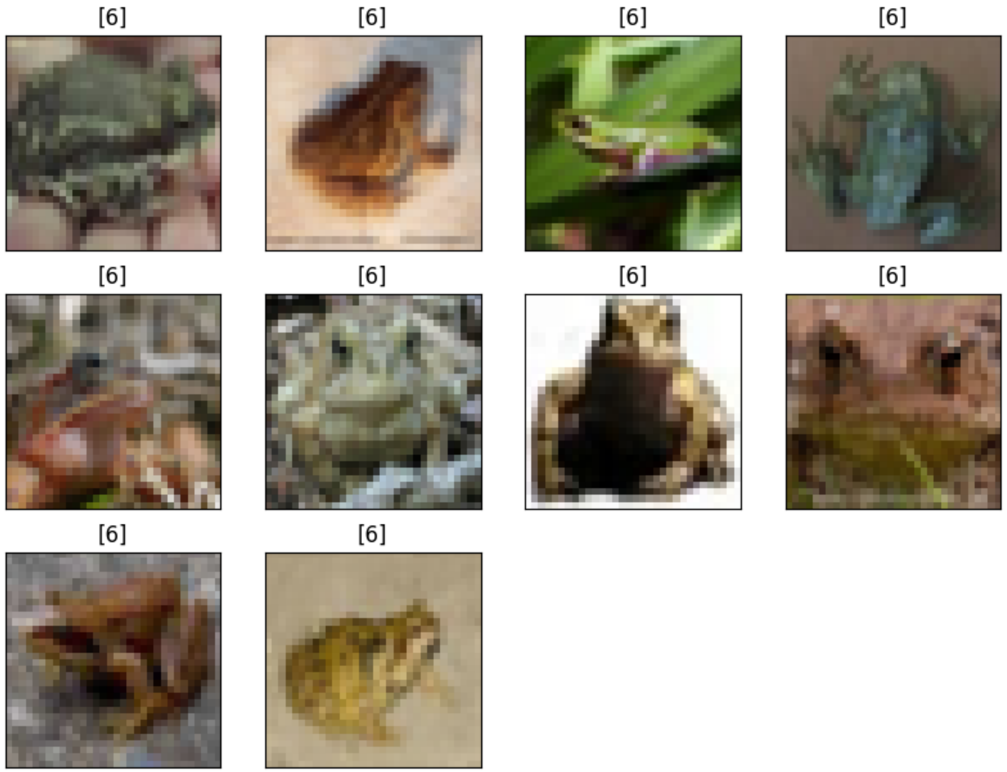
To address this, we need models to be able to "**forget**" specific examples upon demand. At the same time, they need to still perform well on the **remaining** examples. Ideally, we'd also like some proof of the model forgetting certain examples too.

However, this is currently a difficult task to do and to prove, which has led to the emergence of the machine unlearning subfield.

Dataset

Forget Set

examples we want the model to forget

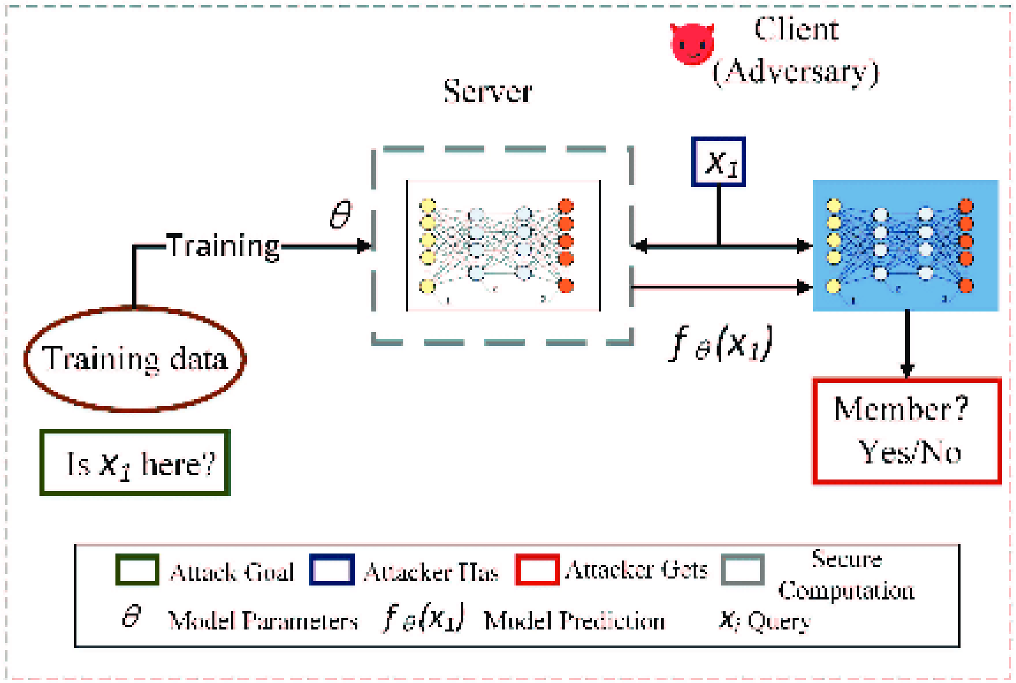


Retain Set

examples we want the model to remember

Forget Guarantee/Membership Inference

- Membership inference attacks are techniques that aim to determine if specific data points were part of the training dataset used to train a model
- These attacks can exploit vulnerabilities in the model's behavior and identify potential privacy risks
- We can instead use this as a metric for our "forget set," to see when our adversarial model is unable to identify our forget set examples in our training data



Existing Solutions

Solution	Limitation
Remove the examples-to-forget (forget set) and retrain the model from scratch with the remaining data (retain set)**	Uses a lot of **time** and **computational resources**
Finetune the model	**No guarantee of forgetting**
Online learning	Cannot be done on-demand, **no guarantee of forgetting**
Gradient Ascent Alongside Finetuning	Limited Results (elaborated later)

**although limited, this is our best scenario for comparing the effectiveness of solution

Approach Overview

Student Teacher Architecture

Prune Stage:

- Sparse models are ideal to work with.
- Find the weights most relevant to the forget set and zero "x%" of them out

Relearn/Finetune Stage:

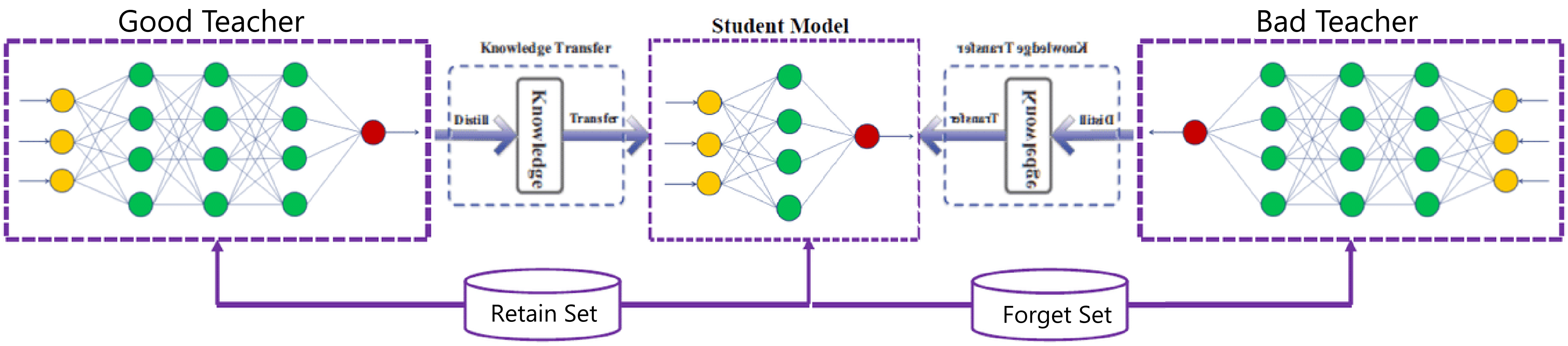
- Retrain on the retain set to make sure it can still accomplish its objective

Forget Stage:

- Force model to continue "unlearning" forget set

Student Teacher Architecture

- Student tries to emulate the teacher model through a loss of KL Divergence
- “Good teacher” provides accurate distribution for retain set while “bad teacher” provides random inputs to further push weights away
- Student aims to minimize the KL divergence between its output and the good teacher’s output while also minimizing the KL divergence between its output and the bad teacher’s output.



Results

Starting Point

"Perfect world"

Metric	Fully Trained	Baseline	Finetuning	No Pruning*	My Approach*
Retain Set Accuracy	0.93	0.8618	0.8730	0.8711	0.7712
Forget Set Accuracy	1.0	0.8999	1.0	0.8614	0.7911
AUC (Membership Inference Attack)	0.93	0.57	0.90	0.71	0.60
Runtime	N/A	467 minutes	40 min/10 epochs	41 minutes	42 minutes

Currently running * for more iterations and hyperparameter variations, so results may change

- Proposed solution's architecture provides the highest guarantee of forgetness while also showing high potential for improved results upon more training.

Conclusion

- Machine unlearning allows for models to be “fixed” if certain parts of its train dataset are discovered to be poisoned and/or must be edited.

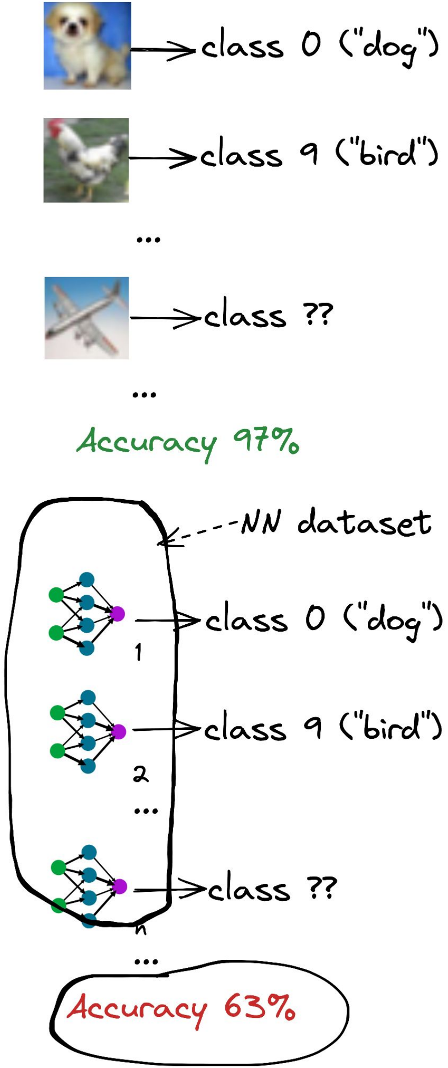
- Has application to many other subfields too (like transfer learning, security)

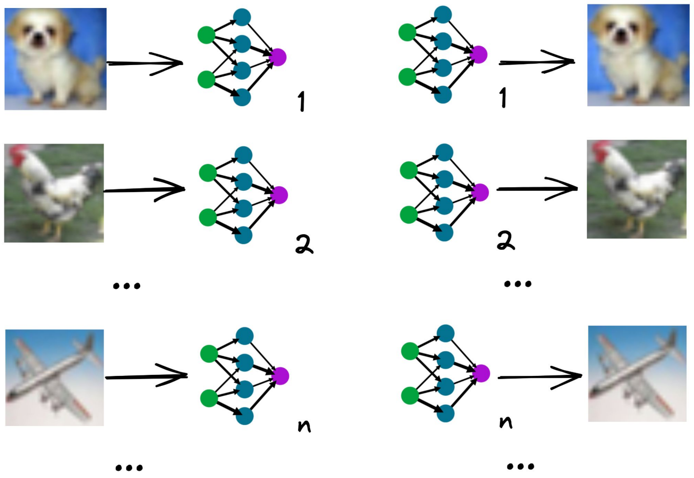
- Still a very under-developed field (most papers published 2023), so lots of ongoing development of new metrics, faster and more effective techniques of unlearning.

- At the end of the day, the work in this subfield is contributing to the development of more adaptive, fair and privacy-aware machine learning systems.

Graph autoencoders
for learning on datasets of neural networks

Alexander Lyzhov



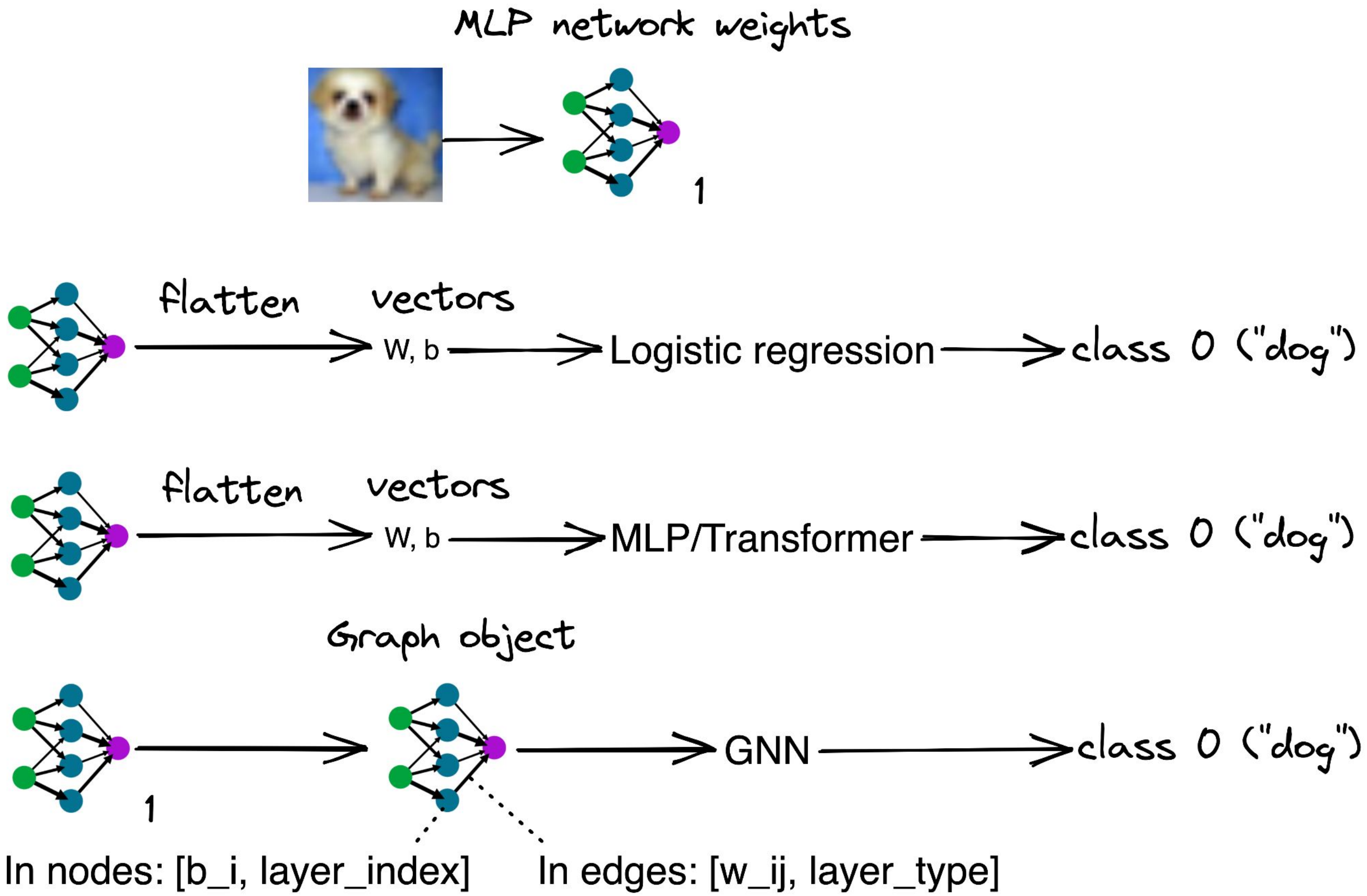


Problem: **classification of neural networks**

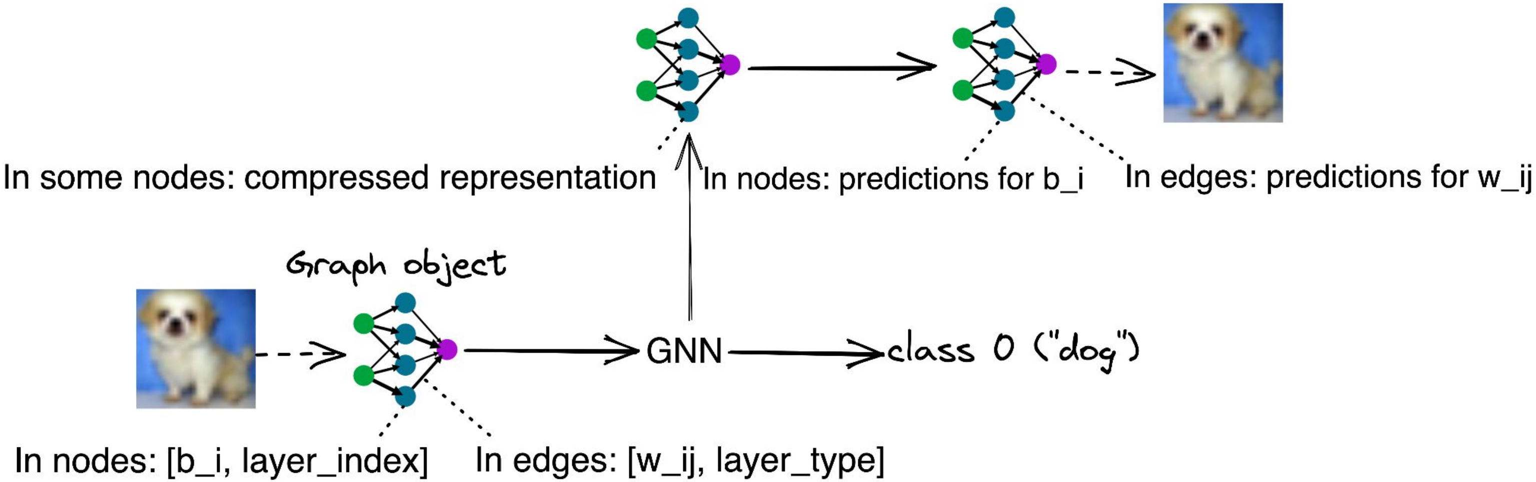
Why solve this problem?
Applications:

- ML on compressed data
- Learning to recognize NN properties
- Learning to edit NNs

Approaches to classification of neural networks



My approach: Relational Attention GNN



Loss = classification loss + reconstruction loss
Options for reconstruction loss:

- L2 in weight space
- L2 in image space

GNN: "Relational Attention", Cameron Diao et al., 2022

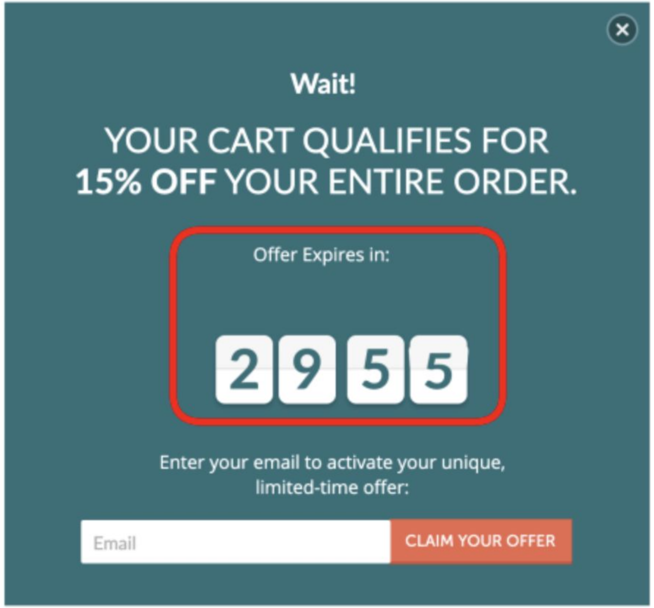
Outcomes:

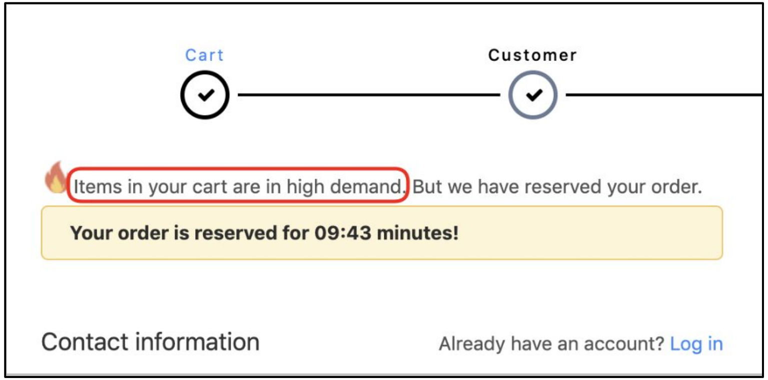
- L2 in weight space is optimized well, but not L2 in image space
- Neither helps classification
- Reason: difficulties with GNN optimization

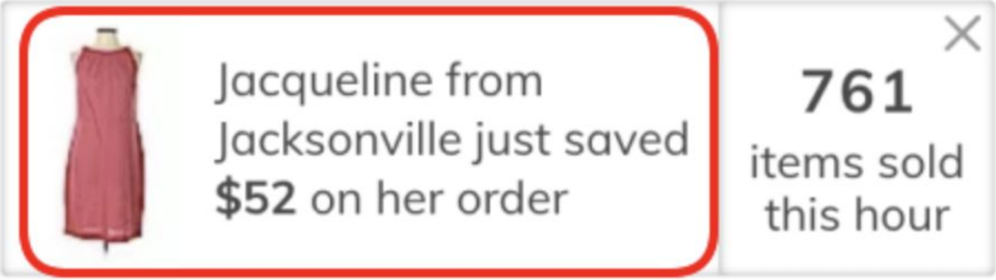
Shedding Light on Dark Patterns

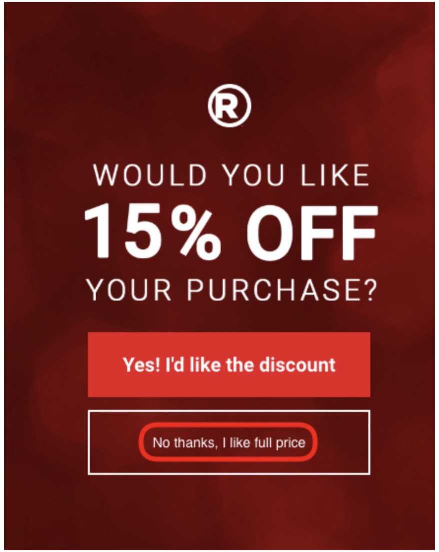
What is a dark pattern?

design elements intentionally crafted to manipulate or deceive users into taking actions they might not want to take or to hinder them from making informed choices


Wait!
YOUR CART QUALIFIES FOR
15% OFF YOUR ENTIRE ORDER.
Offer Expires in:
2 9 5 5
Enter your email to activate your unique,
limited-time offer:
Email
CLAIM YOUR OFFER


Cart
Customer
Items in your cart are in high demand. But we have reserved your order.
Your order is reserved for 09:43 minutes!
Contact information
Already have an account? Log in


Jacqueline from
Jacksonville just saved
$52 on her order
761
items sold
this hour


WOULD YOU LIKE
15% OFF
YOUR PURCHASE?
Yes! I'd like the discount
No thanks, I like full price

“Dark Patterns at Scale: Findings from a Crawl of 11K Shopping Websites”

Mathur et al. (2019)

built a web crawler to visit the ~11K most popular shopping websites worldwide, creating a large data set of dark patterns and documenting their prevalence

Taxonomy of Dark Patterns

URGENCY

Taxonomy
of
Dark Patterns

URGENCY

Taxonomy
of
Dark Patterns

misdire
ction

URGENCY

Taxonomy
of
Dark Patterns

misdire
ction

social proof
social proof social
proof social proof social
proof social proof
social proof social proof

URGENCY

Scarcity

Taxonomy
of
Dark Patterns

social proof
social proof social
proof social proof social
proof social proof
social proof social proof

misdire
ction

“Dark patterns in e-commerce: a dataset and its baseline evaluations”
Yada et al. (2022)

binary classification of dark patterns

Dark patterns in e-commerce: a dataset and its baseline evaluations

Yuki Yada*, Jiaying Feng*, Tsuneo Matsumoto†, Nao Fukushima‡, Fuyuko Kido§¶, Hayato Yamana*
* *Waseda University*, Tokyo, Japan, E-mail: {yada_yuki, kayouh, yamana}@yama.info.waseda.ac.jp
† *National Consumer Affairs Center of Japan*, Sagamihara, Kanagawa, Japan, Email: tsuneo.matsumoto@aoni.waseda.jp
‡ *LINE Corporation*, Shinjuku, Tokyo, Japan, Email: nao.fukushima@linecorp.com
§ *Waseda Research Institute for Science and Engineering*, Tokyo, Japan, Email: fkido@aoni.waseda.jp
¶ *National Institute of Informatics*, Chiyoda-ku, Tokyo, Japan

***Abstract*—Dark patterns, which are user interface designs in online services, induce users to take unintended actions. Recently, dark patterns have been raised as an issue of privacy and fairness. Thus, a wide range of research on detecting dark patterns is eagerly awaited. In this work, we constructed a dataset for dark pattern detection and prepared its baseline detection performance with state-of-the-art machine learning methods. The original dataset was obtained from Mathur et al.'s study in 2019 [1], which consists of 1,818 dark pattern texts from shopping sites. Then, we added negative samples, i.e., non-dark pattern texts, by retrieving texts from the same websites as Mathur et al.'s dataset. We also applied state-of-the-art machine learning methods to show the automatic detection accuracy as baselines, including BERT, RoBERTa, ALBERT, and XLNet. As a result of 5-fold cross-validation, we achieved the highest accuracy of 0.975 with RoBERTa. The dataset and baseline source codes are available at https://github.com/yamanalab/ec-darkpattern.**

***Index Terms*—Dark Patterns, Privacy, User Protection, Deep Learning, Text Classification**

I. Introduction

A. Dark Patterns

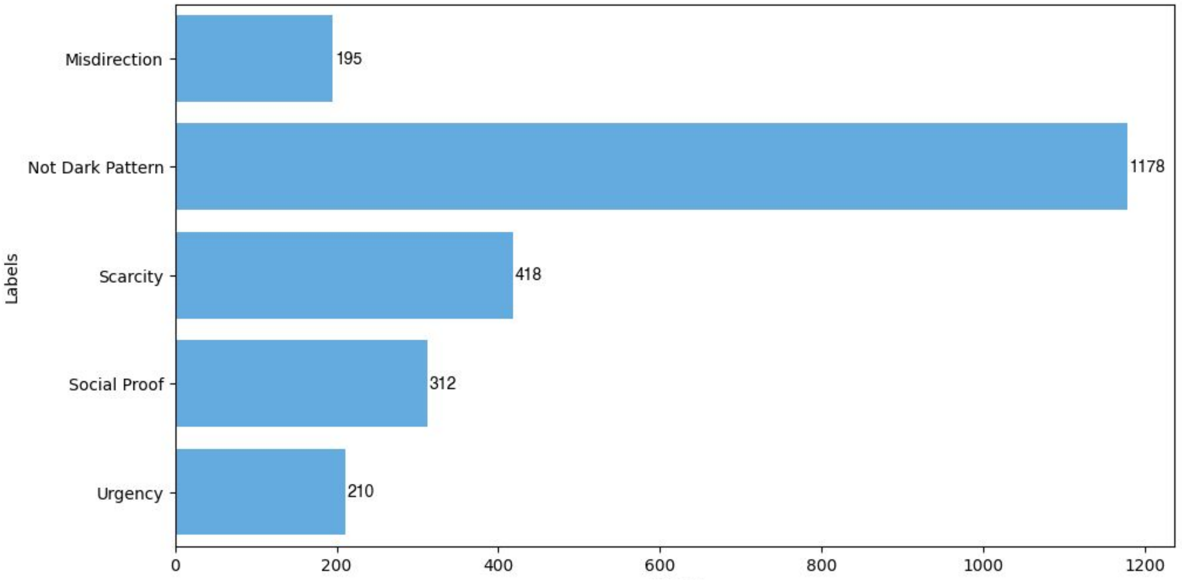
Dark patterns are user interface designs on online services that make users behave in unintended ways. Dark patterns have been called into question in recent years.

In 2010, Harry [2] defined dark patterns as “tricks used in websites and apps that make a user do things that the user did not mean to, like buying or signing up for something.” Fig. 1 shows an example of dark patterns, classified as Obstruction [1]. The obstruction makes it difficult for users to conduct

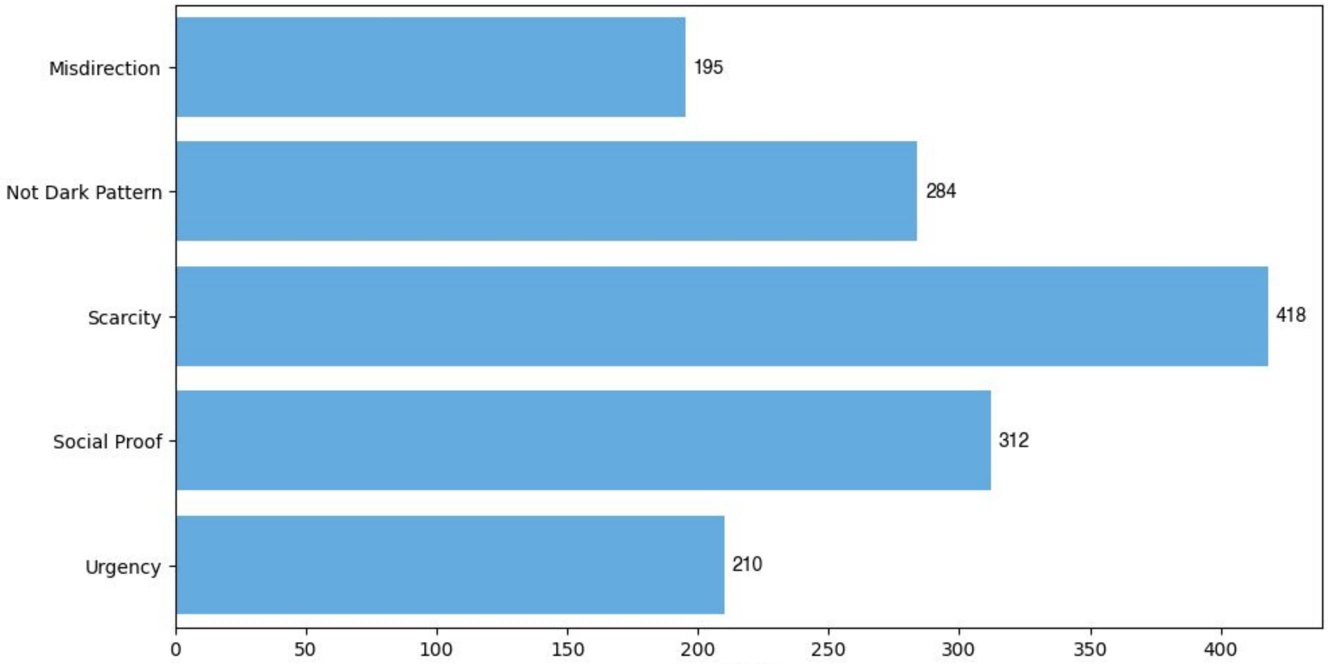
data or consent to cookies in online services. Discussions on the impact of dark patterns to protect user privacy are not limited to academic research and have been widely discussed in various places.

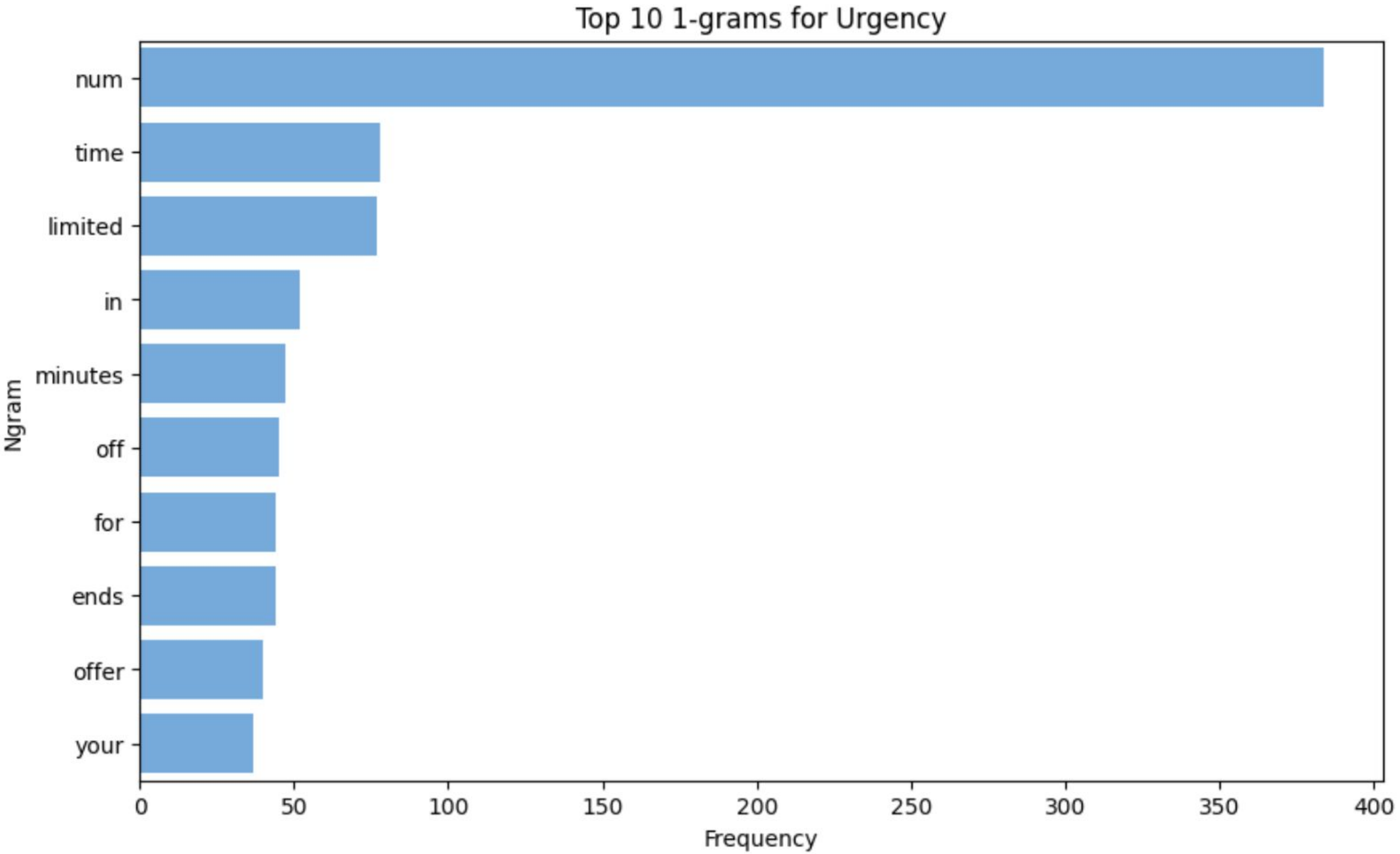
In 2018, the California Legislature passed the Consumer Privacy Protection Act (CCPA) [6] to ban dark patterns on the Internet, which became effective in 2020 and had a critical impact on privacy-related choices. In 2019, the Commission Nationale de l'Informatique et des Libertés (CNIL) in France published a report [7] on the impact of UX design on privacy protection. The report argued that manipulative and/or misleading interfaces on online services could influence critical decisions related to user privacy. The report also raised awareness of such dark patterns and called on designers to collaborate for privacy-friendly designs. In 2020, the Organization for Economic Development and Cooperation (OECD) discussed the privacy and purchasing behavior risks that dark patterns pose to consumers [8]. During the meeting, the risk of dark patterns exposing personal information on online services without the consumer's genuine consent was mentioned. In 2021, the European Data Protection Board (EDPB) discussed dark patterns in social media that can negatively impact users' decisions regarding the handling of personal information [9]. The main objective was to discuss protecting users from dark patterns that may harm their privacy.

As ever-increasing dark patterns have become a social problem, as evidenced by the policies of various countries, *"The original dataset was obtained from Mathur et al.'s study in 2019, which consists of 1,818 dark pattern texts from shopping sites. Then, we added negative samples, i.e., non-dark pattern texts, by retrieving texts from the same websites as Mathur et al.'s dataset."*

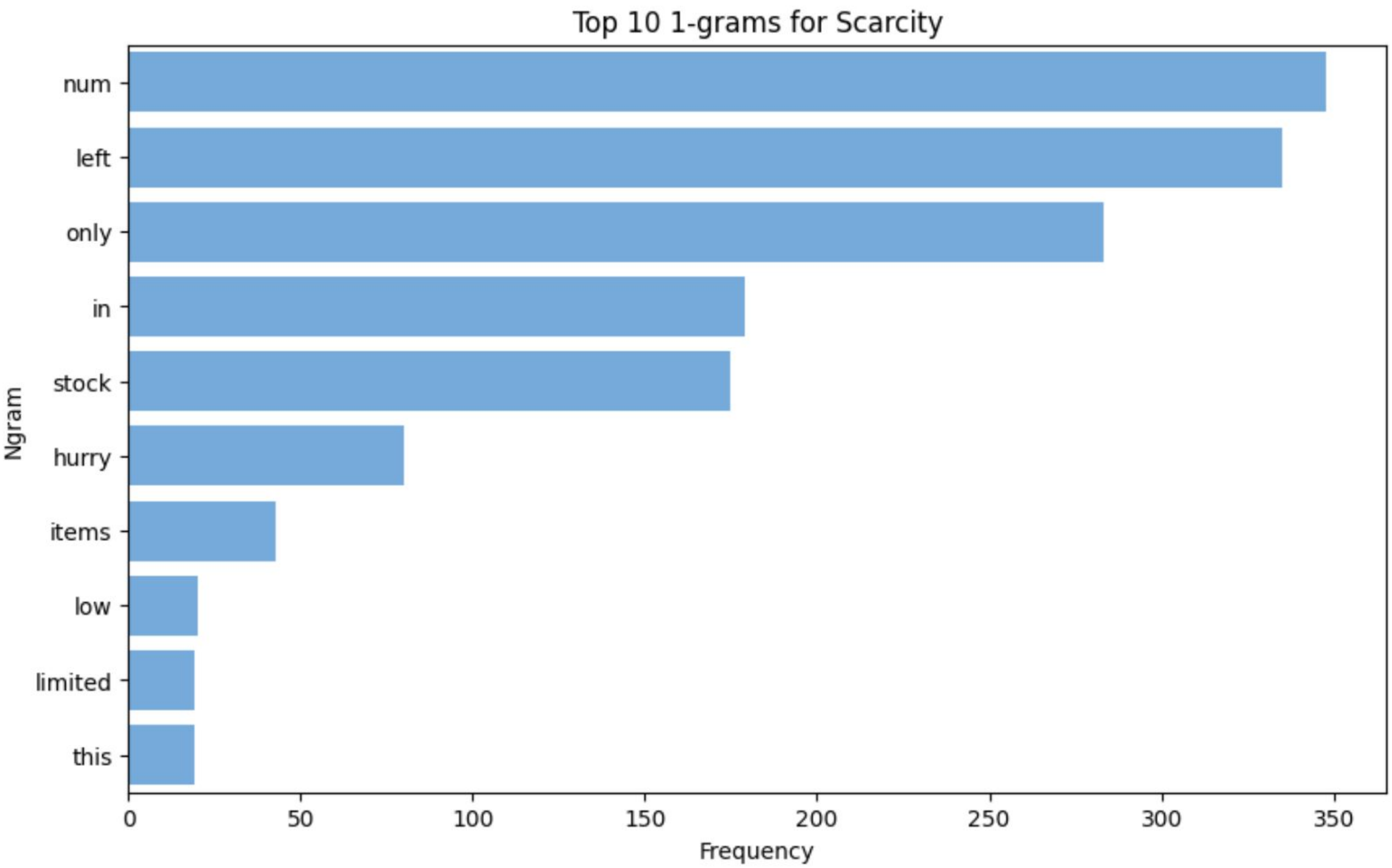


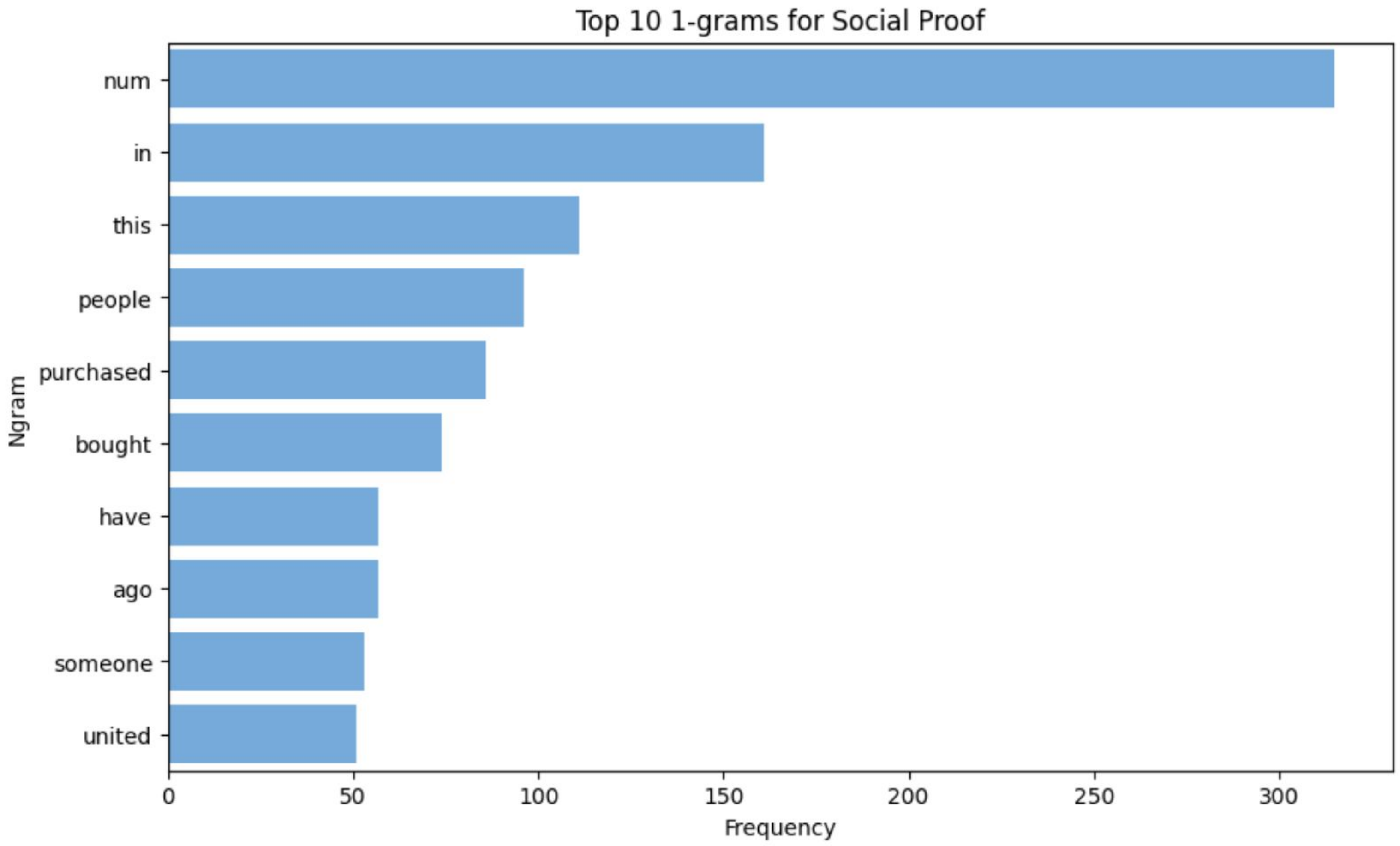
undersampling + text preprocessing =



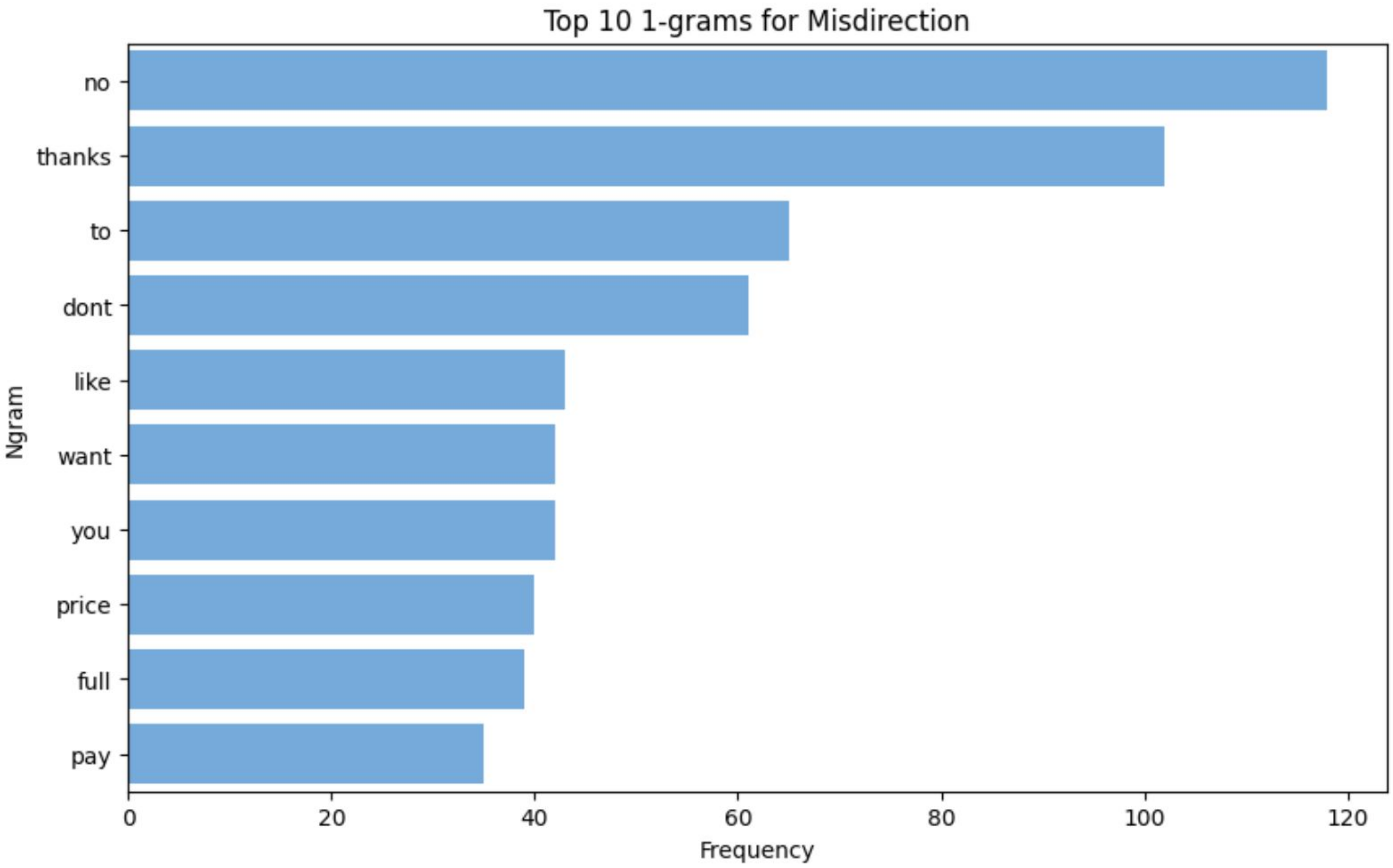


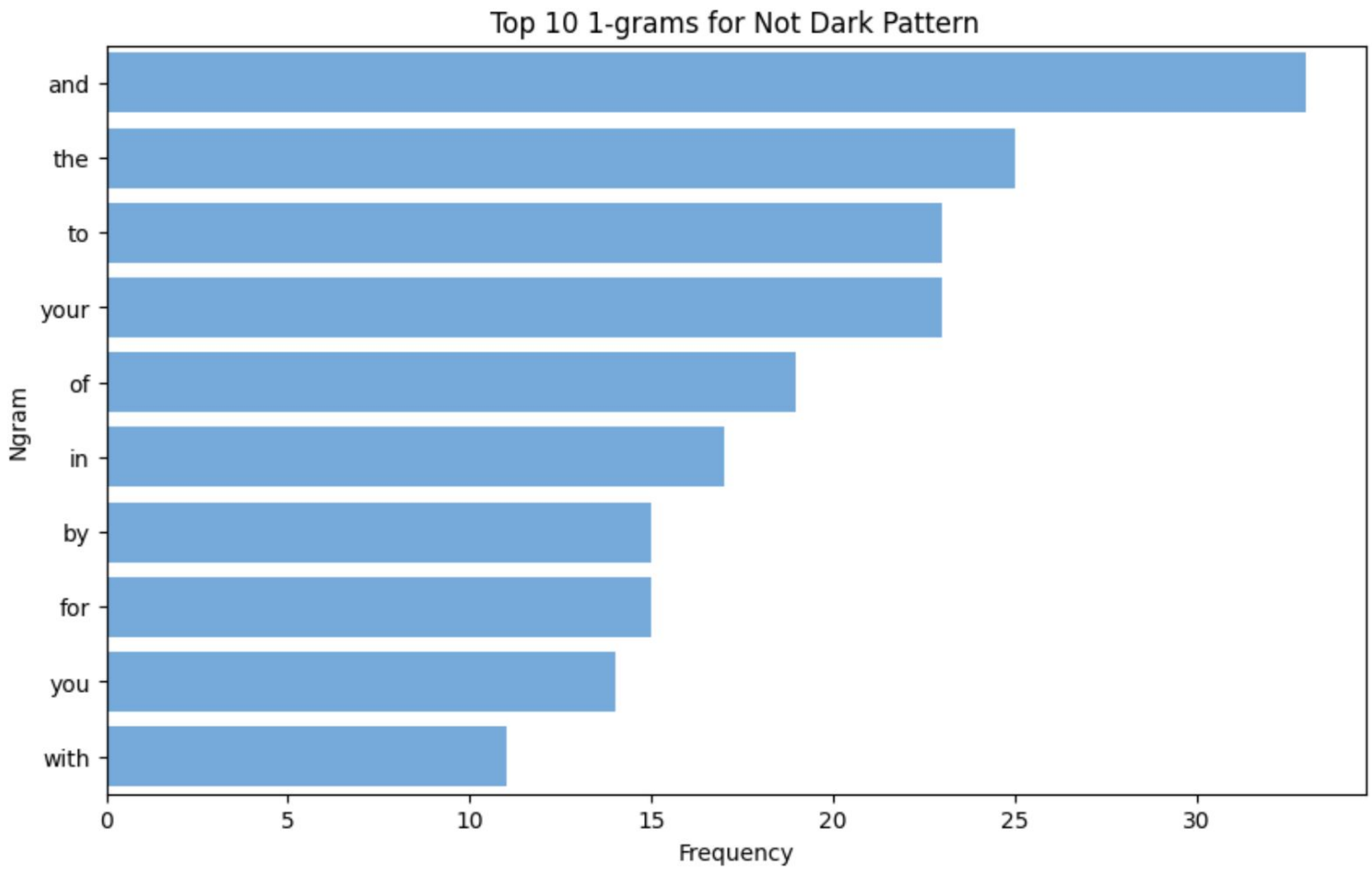
Top 10 1-grams for Urgency
num
time
limited
in
minutes
off
for
ends
offer
your
Ngram
0
50
100
150
200
250
300
350
400
Frequency


Top 10 1-grams for Scarcity
num
left
only
in
stock
hurry
items
low
limited
this
Ngram
0
50
100
150
200
250
300
350
Frequency



Top 10 1-grams for Social Proof
num
in
this
people
purchased
bought
have
ago
someone
united
Ngram
0
50
100
150
200
250
300
Frequency


Top 10 1-grams for Misdirection
no
thanks
to
dont
like
want
you
price
full
pay
Ngram
0
20
40
60
80
100
120
Frequency



Top 10 1-grams for Not Dark Pattern
and
the
to
your
of
in
by
for
you
with
Ngram
0
5
10
15
20
25
30
Frequency

Bag-of-Words

or

Term Frequency
x
Inverse
Document
Frequency

Bag-of-Words

or

Term Frequency
x
Inverse
Document
Frequency

Model Candidates

Naive Bayes

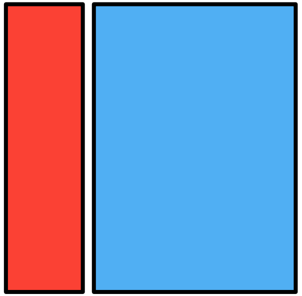
SVM

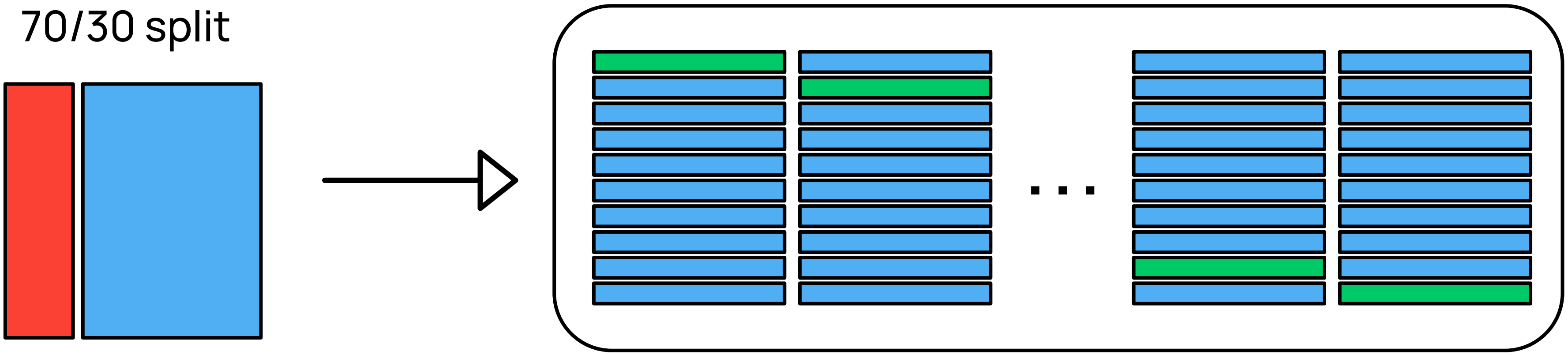
Logistic Regression

Random Forest

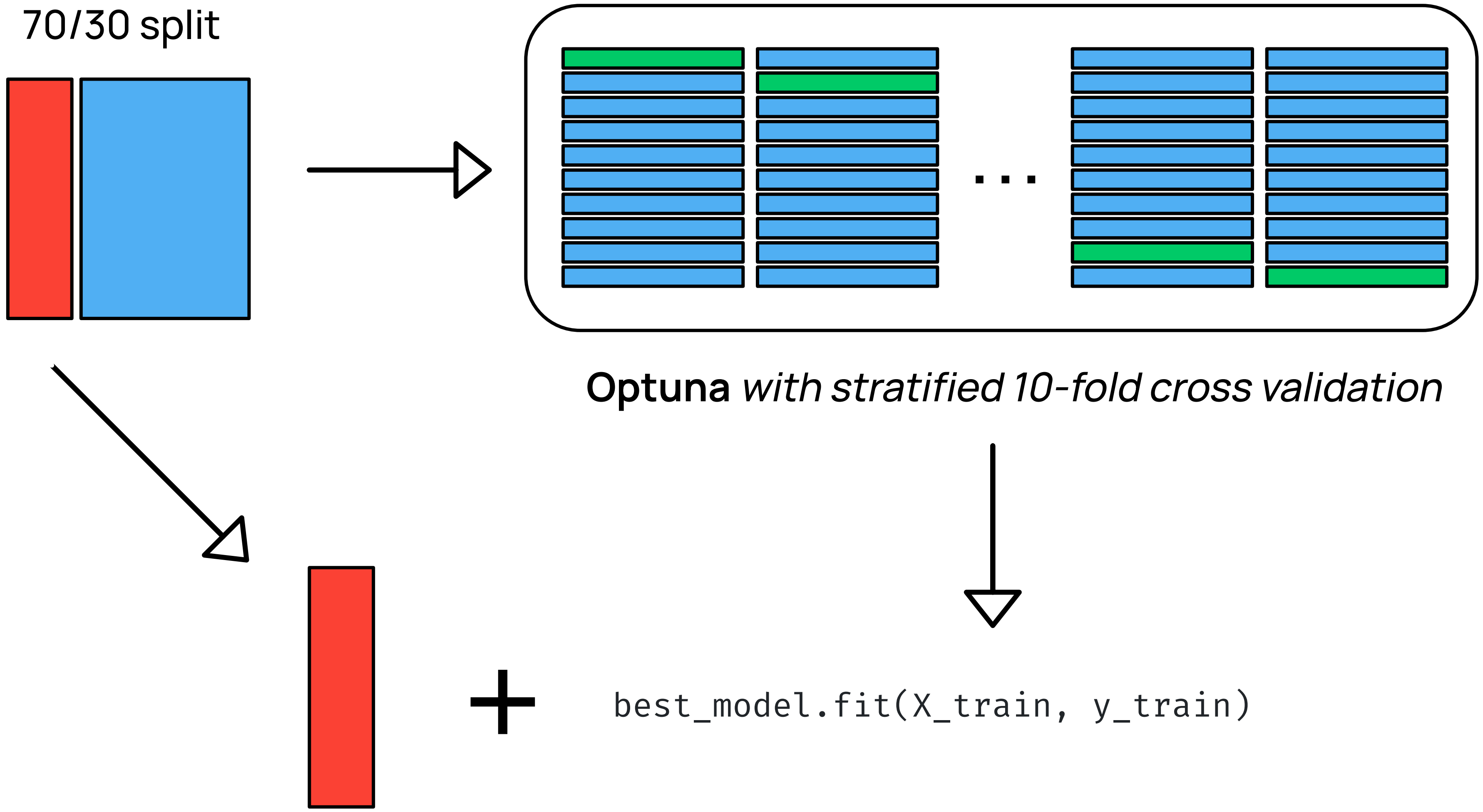
XGBoost

70/30 split





Optuna *with stratified 10-fold cross validation*



70/30 split
...
Optuna *with stratified 10-fold cross validation*
+
best_model.fit(X_train, y_train)

Precision = True Positive / All Positive Predictions

Recall = True Positives / Actual Positive Cases

F1 = 2 x Precision x Recall / (Precision + Recall)

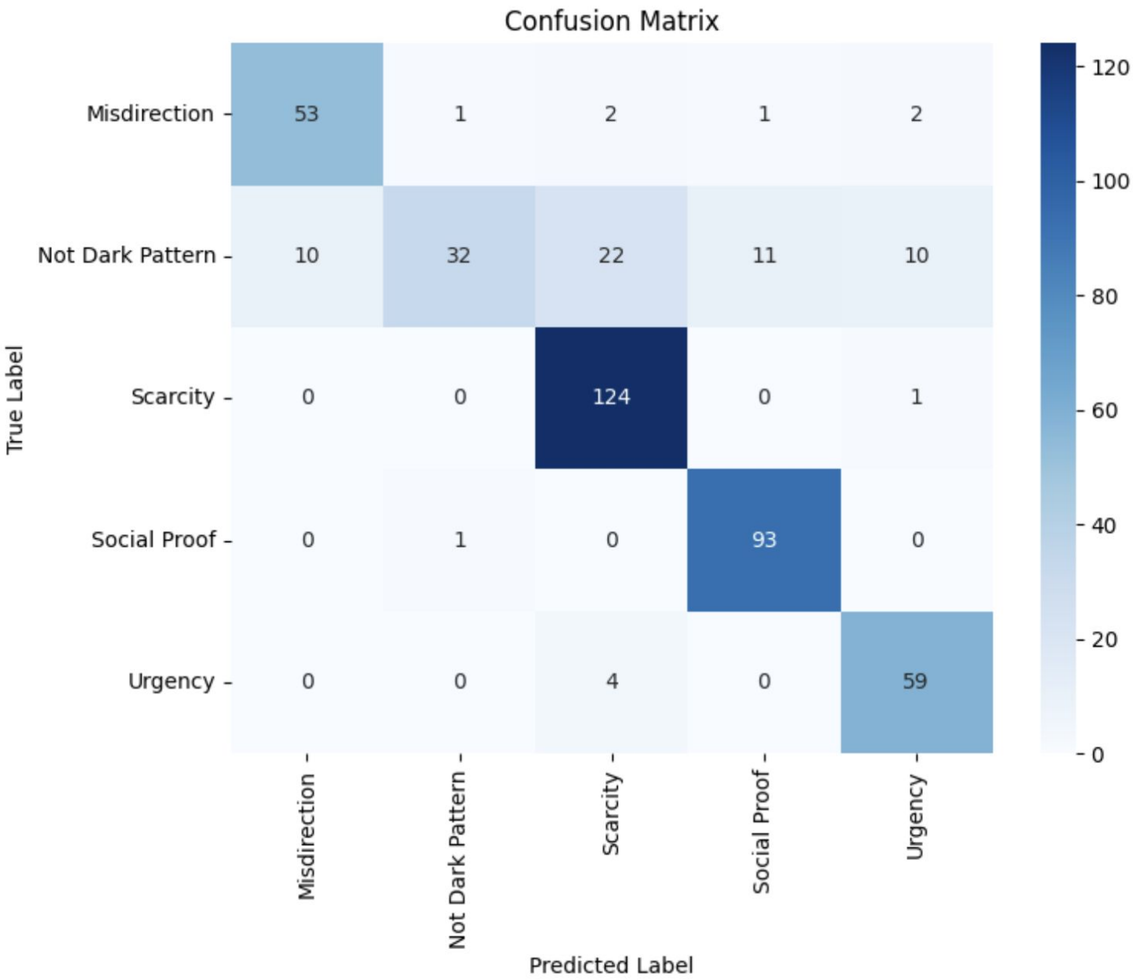
Weighted F1 = Weighted Average of Class F1 Scores

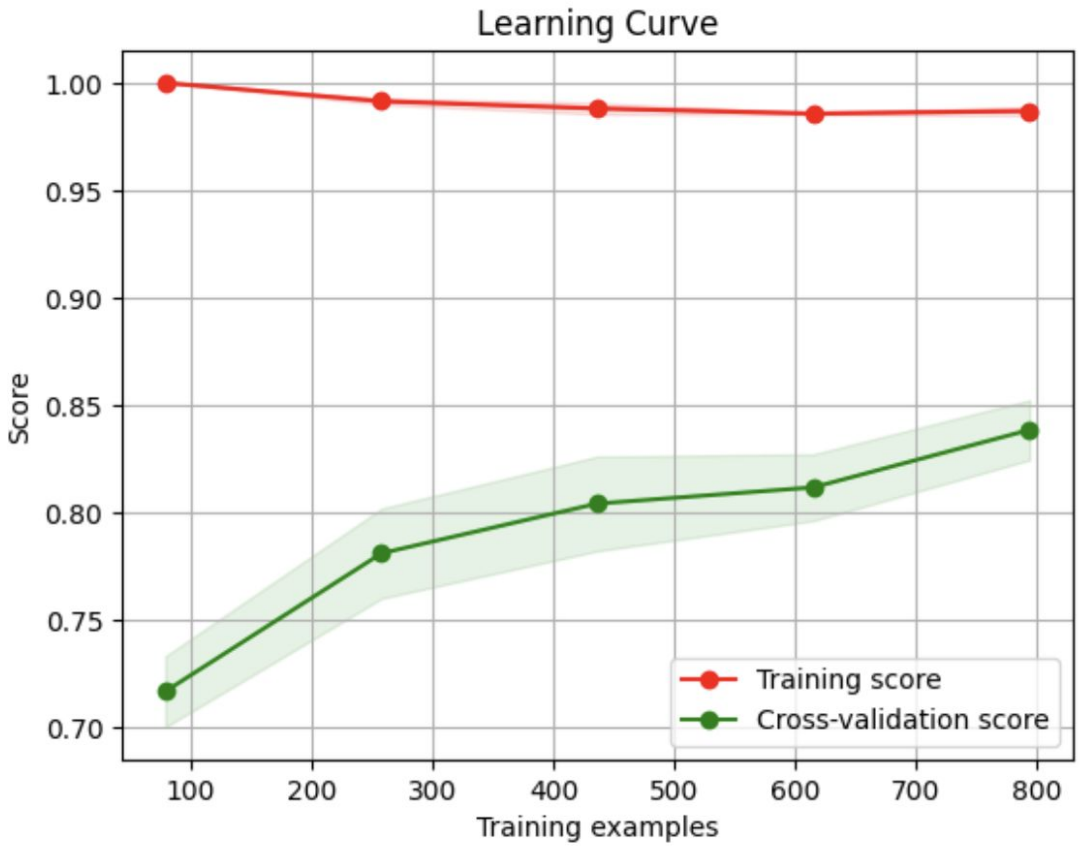
Multinomial Naive Bayes

Accuracy: 0.8474

F1 Score: 0.8259

	precision	recall	f1-score	support
Misdirection	0.84	0.90	0.87	59
Not Dark Pattern	0.94	0.38	0.54	85
Scarcity	0.82	0.99	0.90	125
Social Proof	0.89	0.99	0.93	94
Urgency	0.82	0.94	0.87	63
accuracy			0.85	426
macro avg	0.86	0.84	0.82	426
weighted avg	0.86	0.85	0.83	426


Confusion Matrix
Misdirection
Not Dark Pattern
Scarcity
Social Proof
Urgency
53 1 2 1 2
10 32 22 11 10
0 0 124 0 1
0 1 0 93 0
0 0 4 0 59
True Label
Predicted Label
0
20
40
60
80
100
120

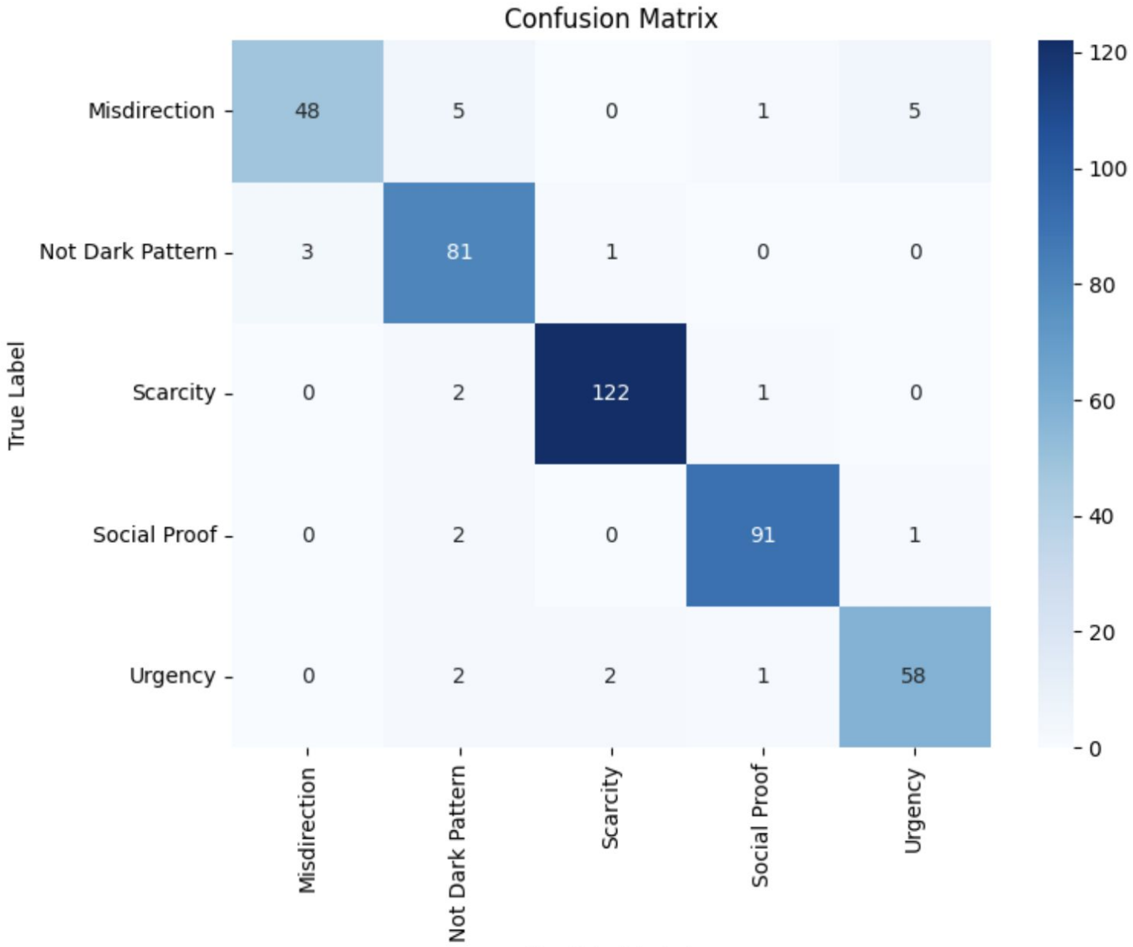

Learning Curve
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Score
100
200
300
400
500
600
700
800
Training examples
Training score
Cross-validation score

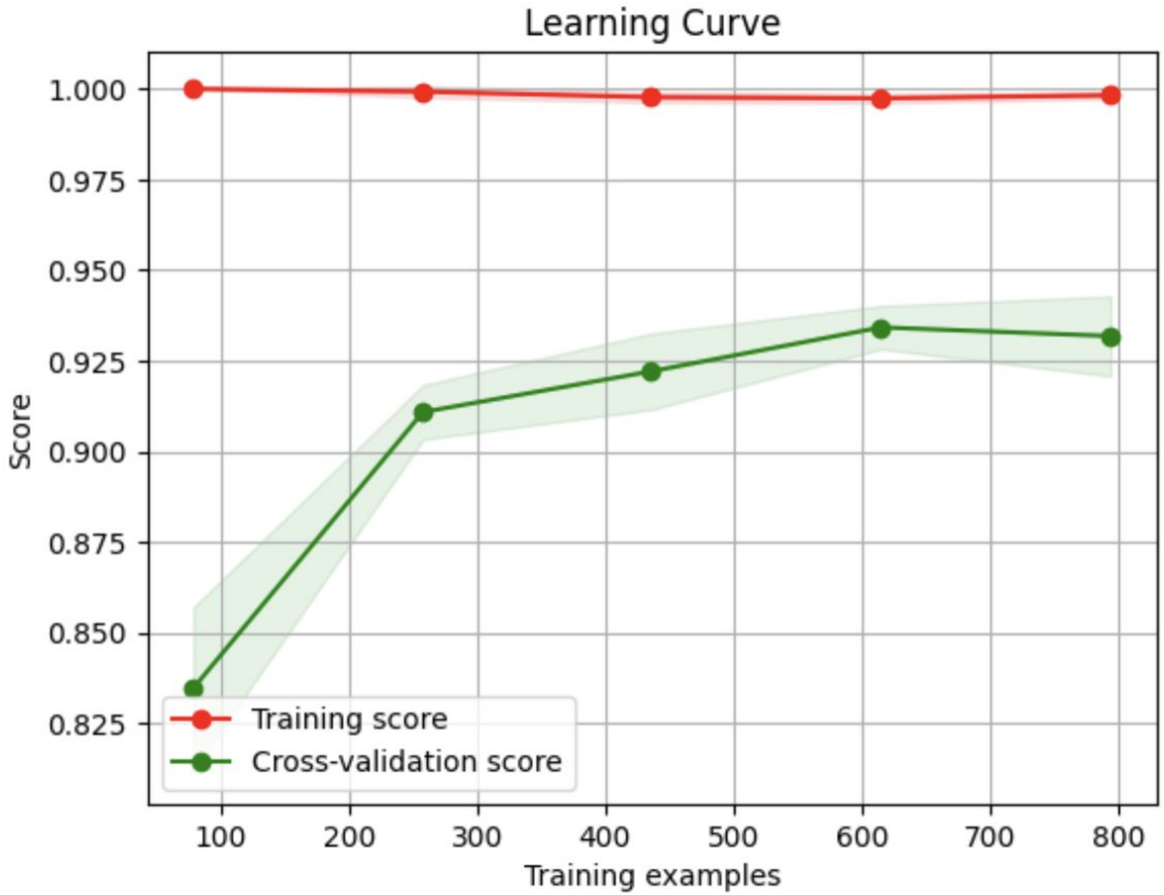
Multinomial L2 Logistic Regression

Accuracy: 0.9390

F1 Score: 0.9396

	precision	recall	f1-score	support
Misdirection	0.94	0.81	0.87	59
Not Dark Pattern	0.88	0.95	0.92	85
Scarcity	0.98	0.98	0.98	125
Social Proof	0.97	0.97	0.97	94
Urgency	0.91	0.92	0.91	63
accuracy			0.94	426
macro avg	0.93	0.93	0.93	426
weighted avg	0.94	0.94	0.94	426


Confusion Matrix
True Label
Predicted Label
Misdirection
Not Dark Pattern
Scarcity
Social Proof
Urgency
48
5
0
1
5
3
81
1
0
0
0
2
122
1
0
0
2
0
91
1
0
2
2
1
58
0
20
40
60
80
100
120



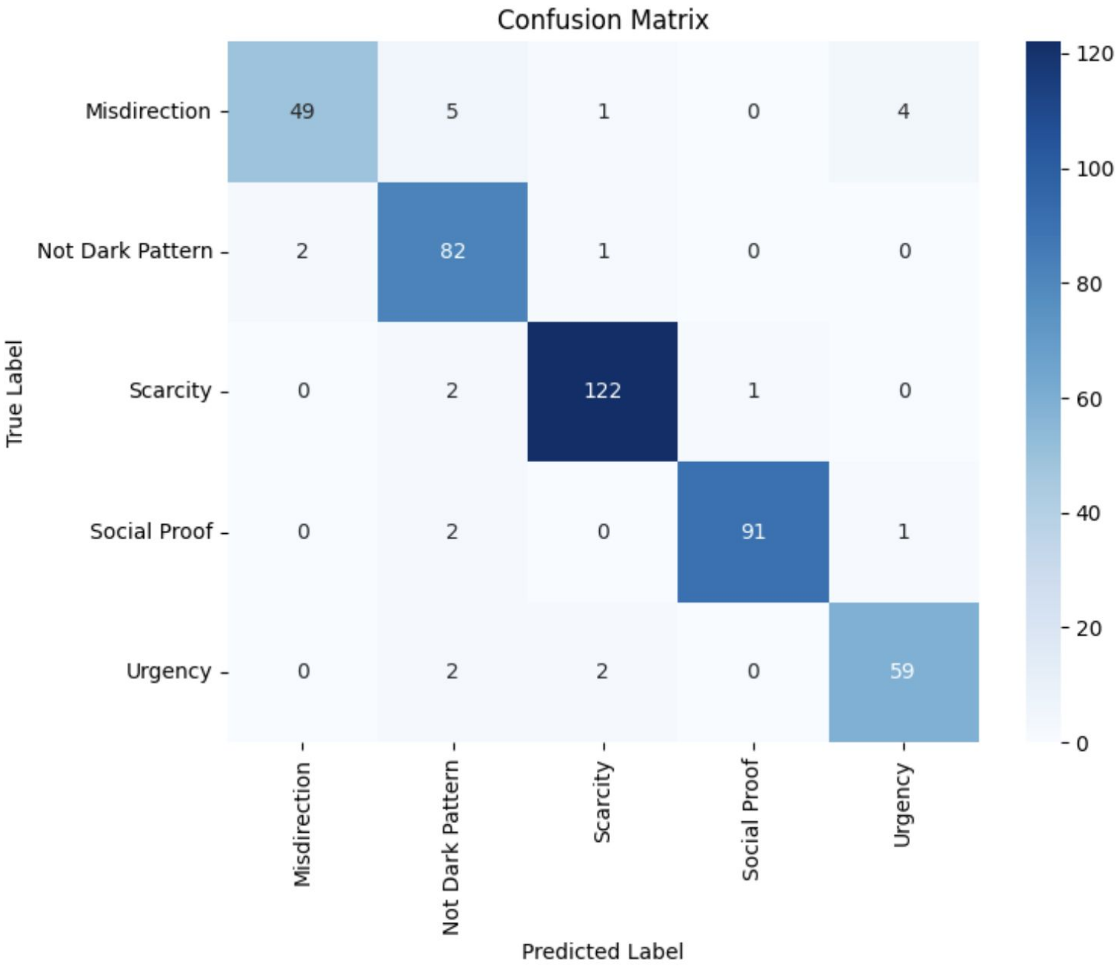
Learning Curve
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Score
100
200
300
400
500
600
700
800
Training examples
Training score
Cross-validation score

SVM with Linear Kernel

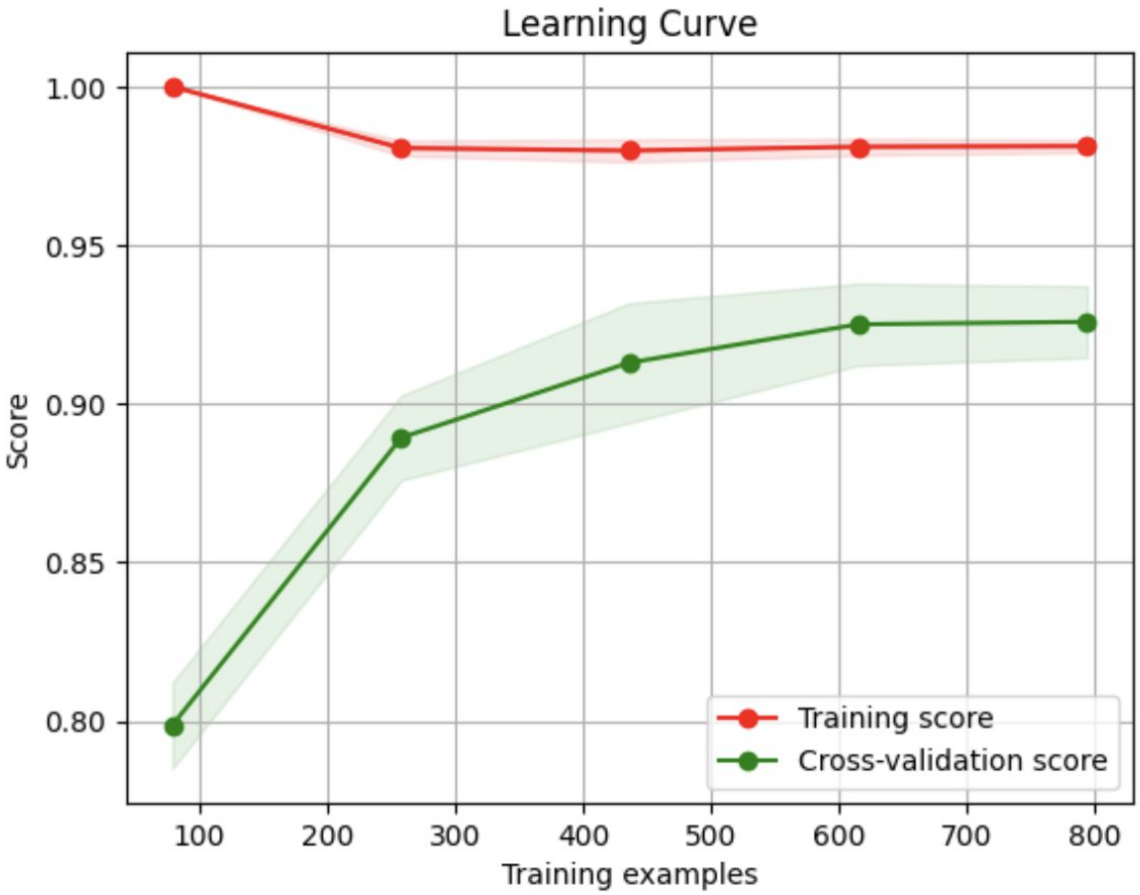
Accuracy: 0.9460

F1 Score: 0.9458

	precision	recall	f1-score	support
Misdirection	0.96	0.83	0.89	59
Not Dark Pattern	0.88	0.96	0.92	85
Scarcity	0.97	0.98	0.97	125
Social Proof	0.99	0.97	0.98	94
Urgency	0.92	0.94	0.93	63
accuracy			0.95	426
macro avg	0.94	0.94	0.94	426
weighted avg	0.95	0.95	0.95	426



Confusion Matrix
True Label
Predicted Label
Misdirection
Not Dark Pattern
Scarcity
Social Proof
Urgency
49
5
1
0
4
2
82
1
0
0
0
2
122
1
0
0
2
0
91
1
0
2
2
0
59
120
100
80
60
40
20
0



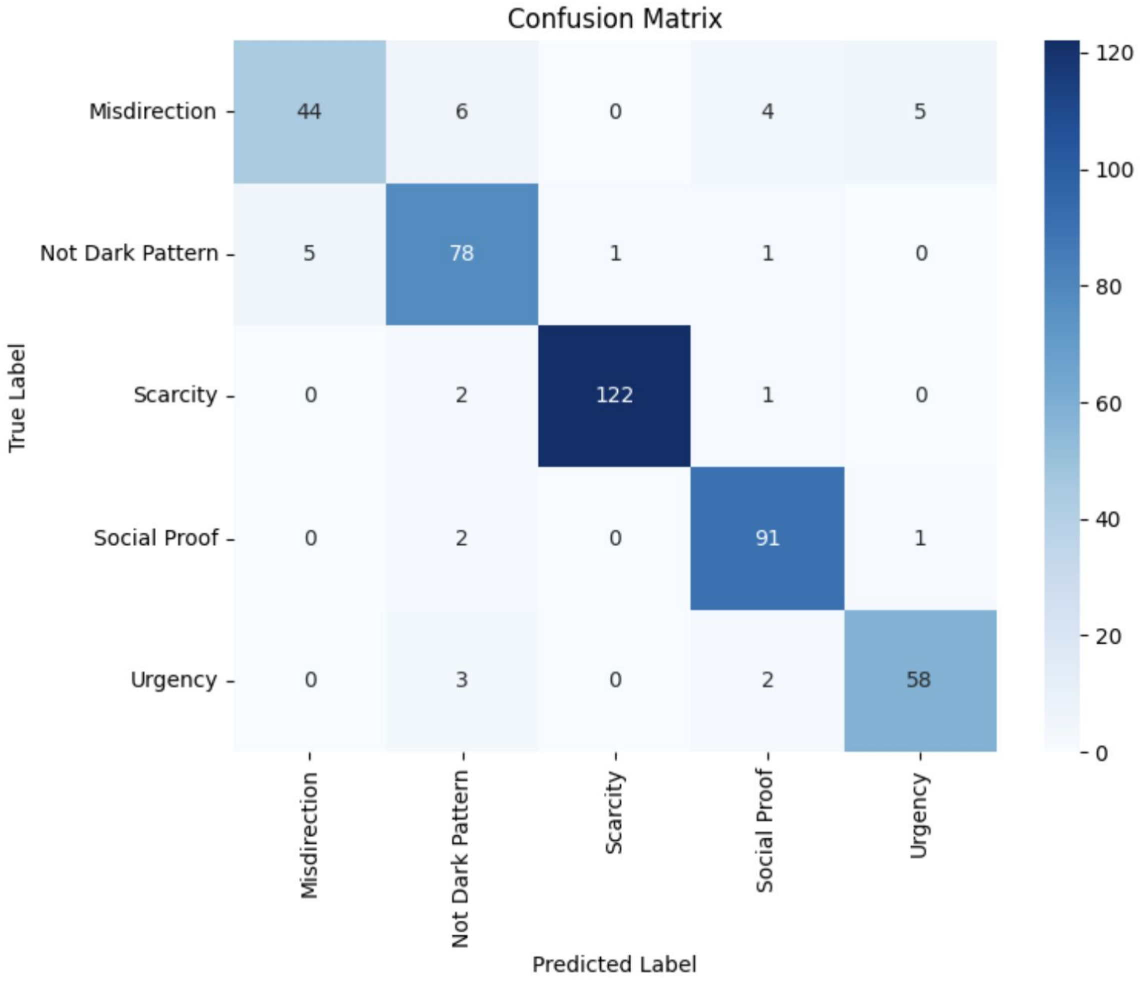
Learning Curve
1.00
0.95
0.90
0.85
0.80
Score
100
200
300
400
500
600
700
800
Training examples
Training score
Cross-validation score

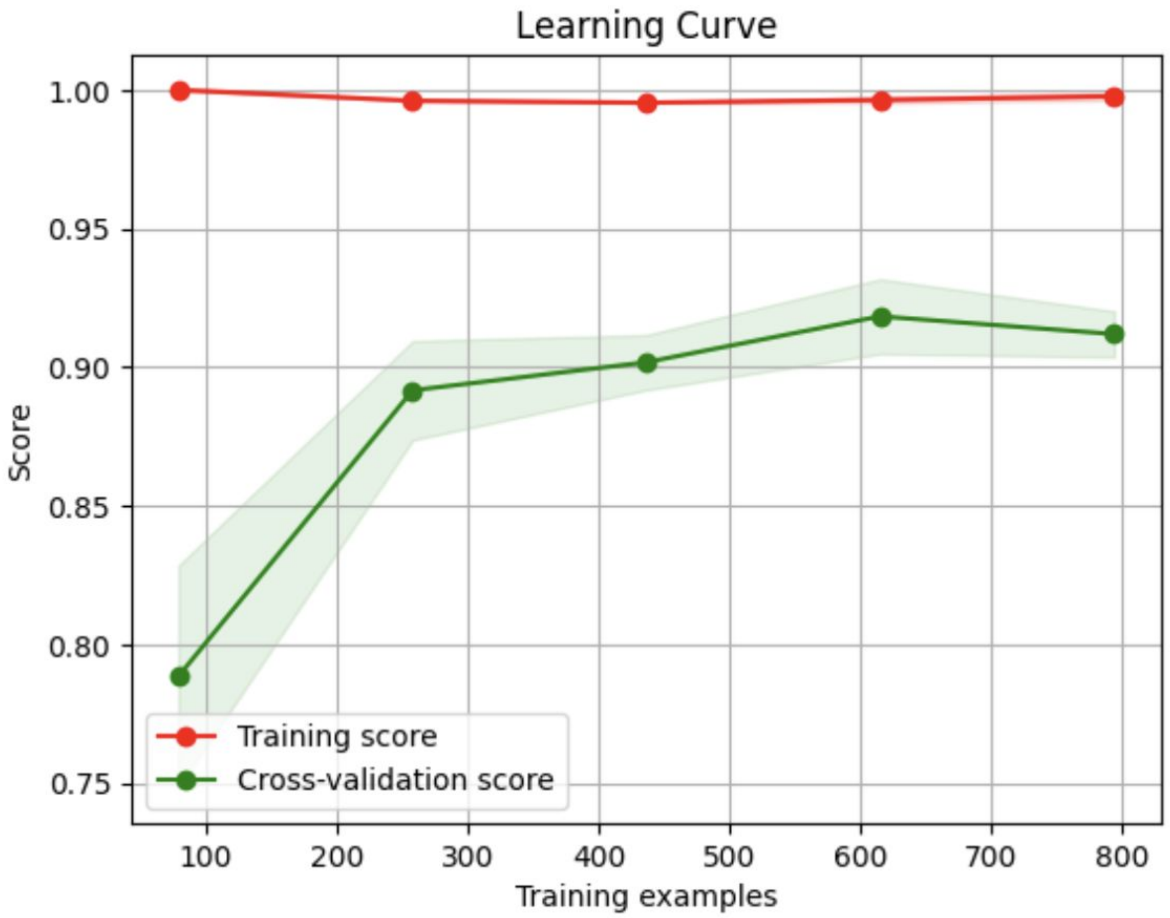
SVM with RBF Kernel

Accuracy: 0.9225

F1 Score: 0.9216

	precision	recall	f1-score	support
Misdirection	0.90	0.75	0.81	59
Not Dark Pattern	0.86	0.92	0.89	85
Scarcity	0.99	0.98	0.98	125
Social Proof	0.92	0.97	0.94	94
Urgency	0.91	0.92	0.91	63
accuracy			0.92	426
macro avg	0.91	0.91	0.91	426
weighted avg	0.92	0.92	0.92	426


Confusion Matrix
Misdirection
Not Dark Pattern
Scarcity
Social Proof
Urgency
True Label
Predicted Label
44
6
0
4
5
5
78
1
1
0
0
2
122
1
0
0
2
0
91
1
0
3
0
2
58
120
100
80
60
40
20
0

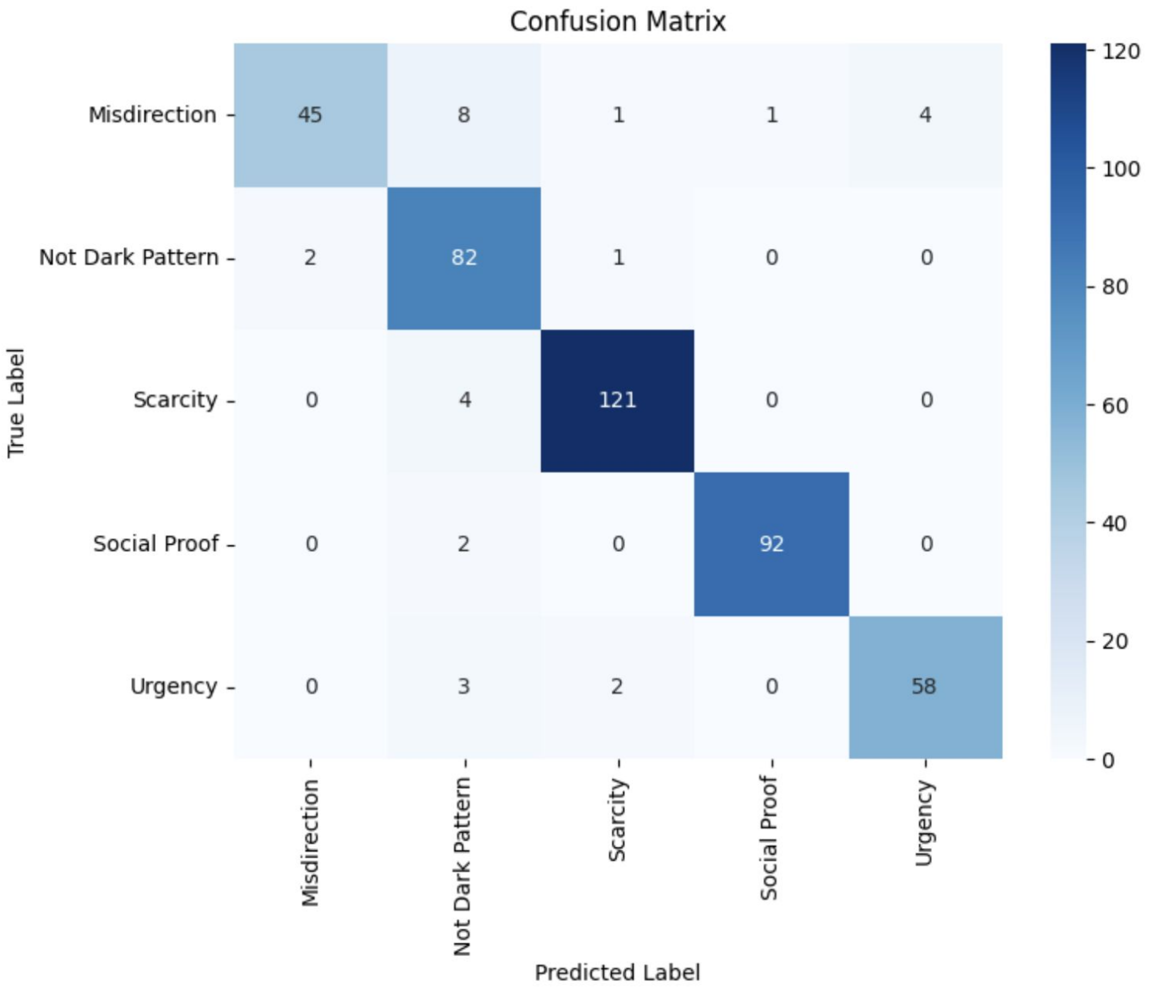

Learning Curve
Score
Training examples
1.00
0.95
0.90
0.85
0.80
0.75
100
200
300
400
500
600
700
800
Training score
Cross-validation score

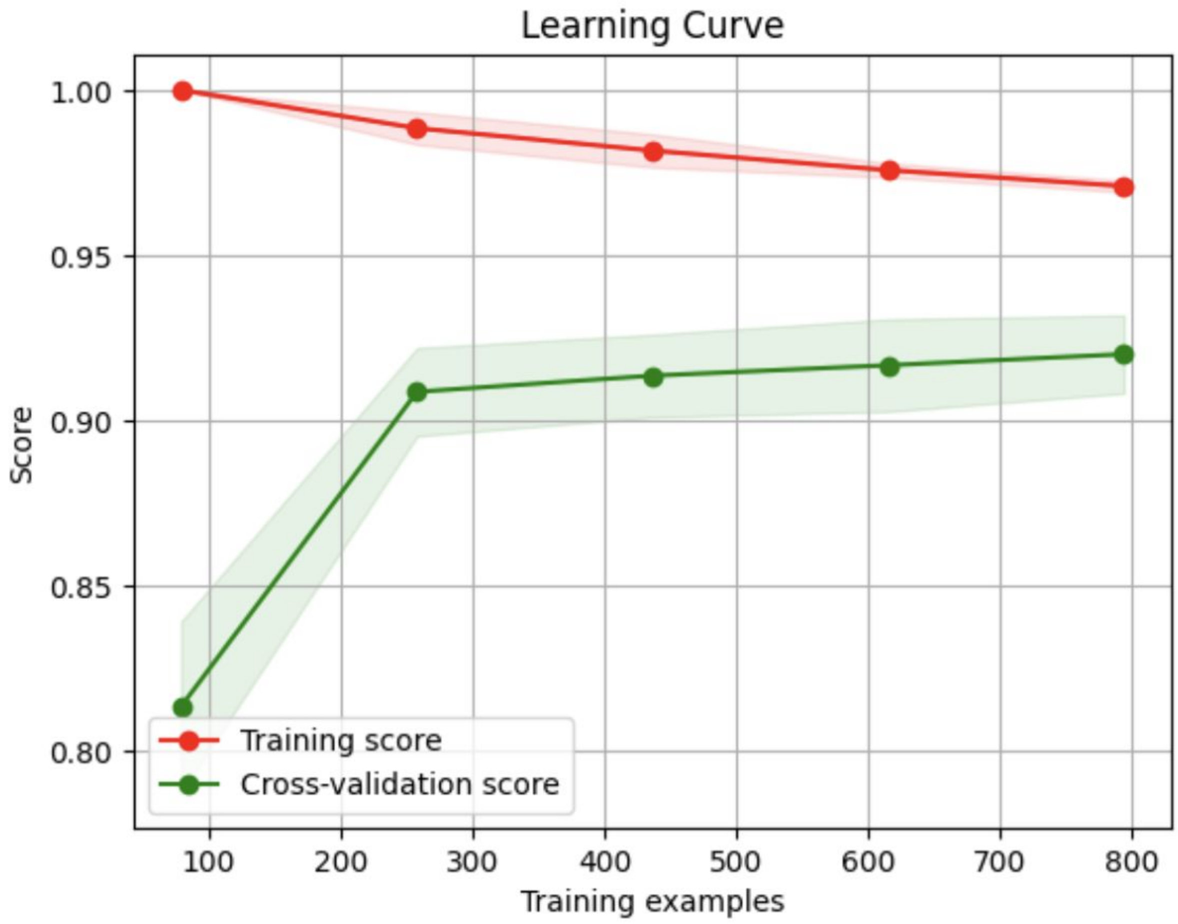
Random Forest

Accuracy: 0.9343

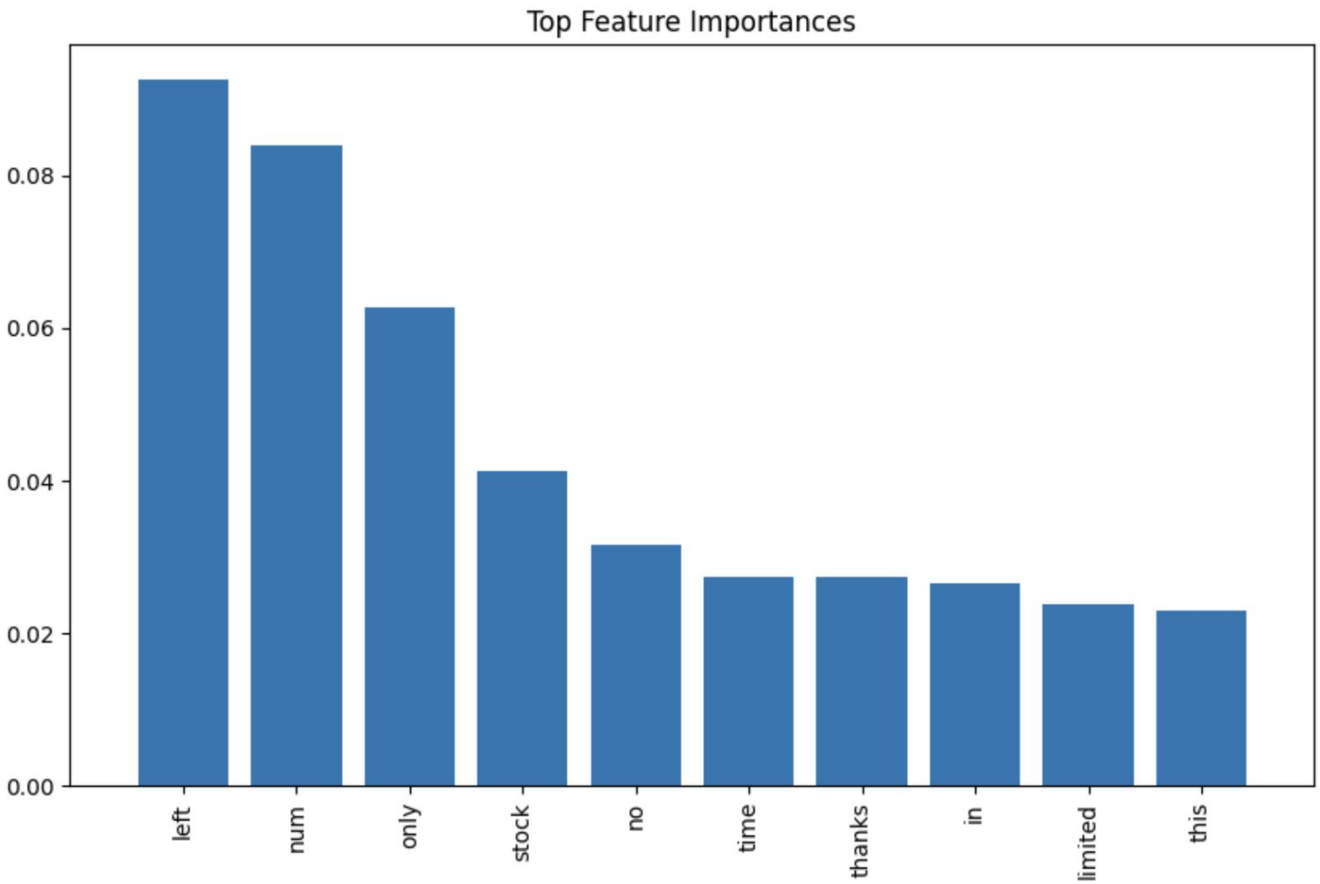
F1 Score: 0.9338

	precision	recall	f1-score	support
Misdirection	0.96	0.76	0.85	59
Not Dark Pattern	0.83	0.96	0.89	85
Scarcity	0.97	0.97	0.97	125
Social Proof	0.99	0.98	0.98	94
Urgency	0.94	0.92	0.93	63
accuracy			0.93	426
macro avg	0.94	0.92	0.92	426
weighted avg	0.94	0.93	0.93	426


Confusion Matrix
True Label
Predicted Label
Misdirection
Not Dark Pattern
Scarcity
Social Proof
Urgency
45
8
1
1
4
2
82
1
0
0
0
4
121
0
0
0
2
0
92
0
0
3
2
0
58
0
20
40
60
80
100
120



Learning Curve
1.00
0.95
0.90
0.85
0.80
Score
100
200
300
400
500
600
700
800
Training examples
Training score
Cross-validation score

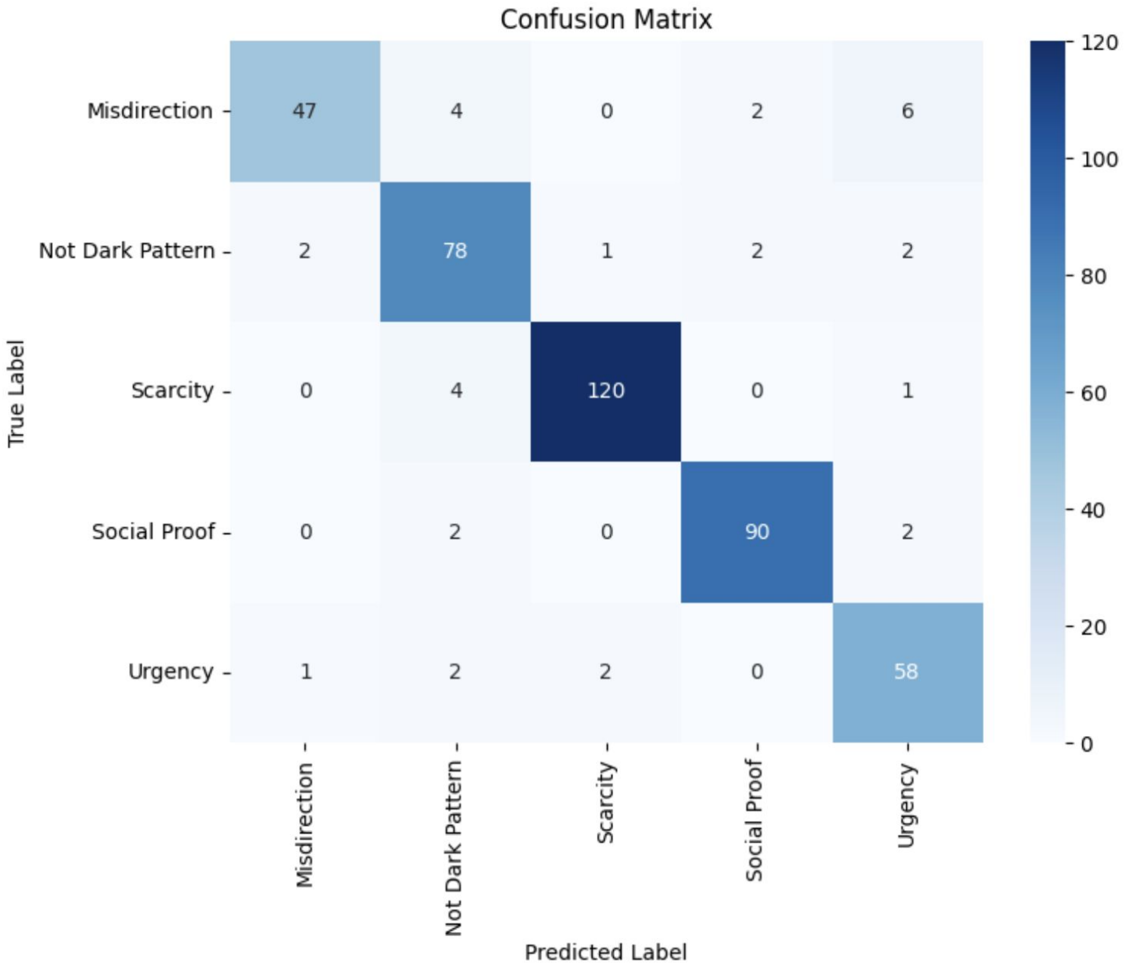

Top Feature Importances
0.00
0.02
0.04
0.06
0.08
left
num
only
stock
no
time
thanks
in
limited
this

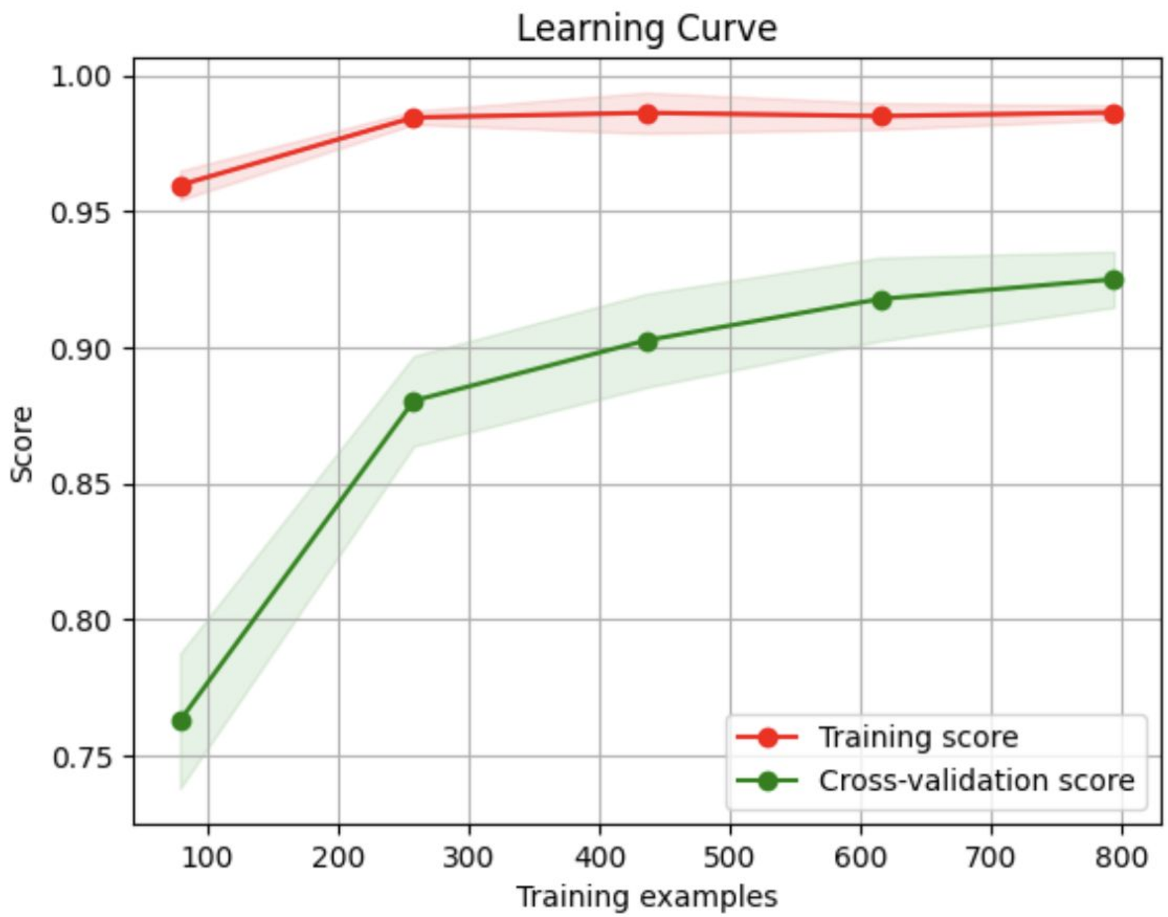
XGBoost

Accuracy: 0.9225

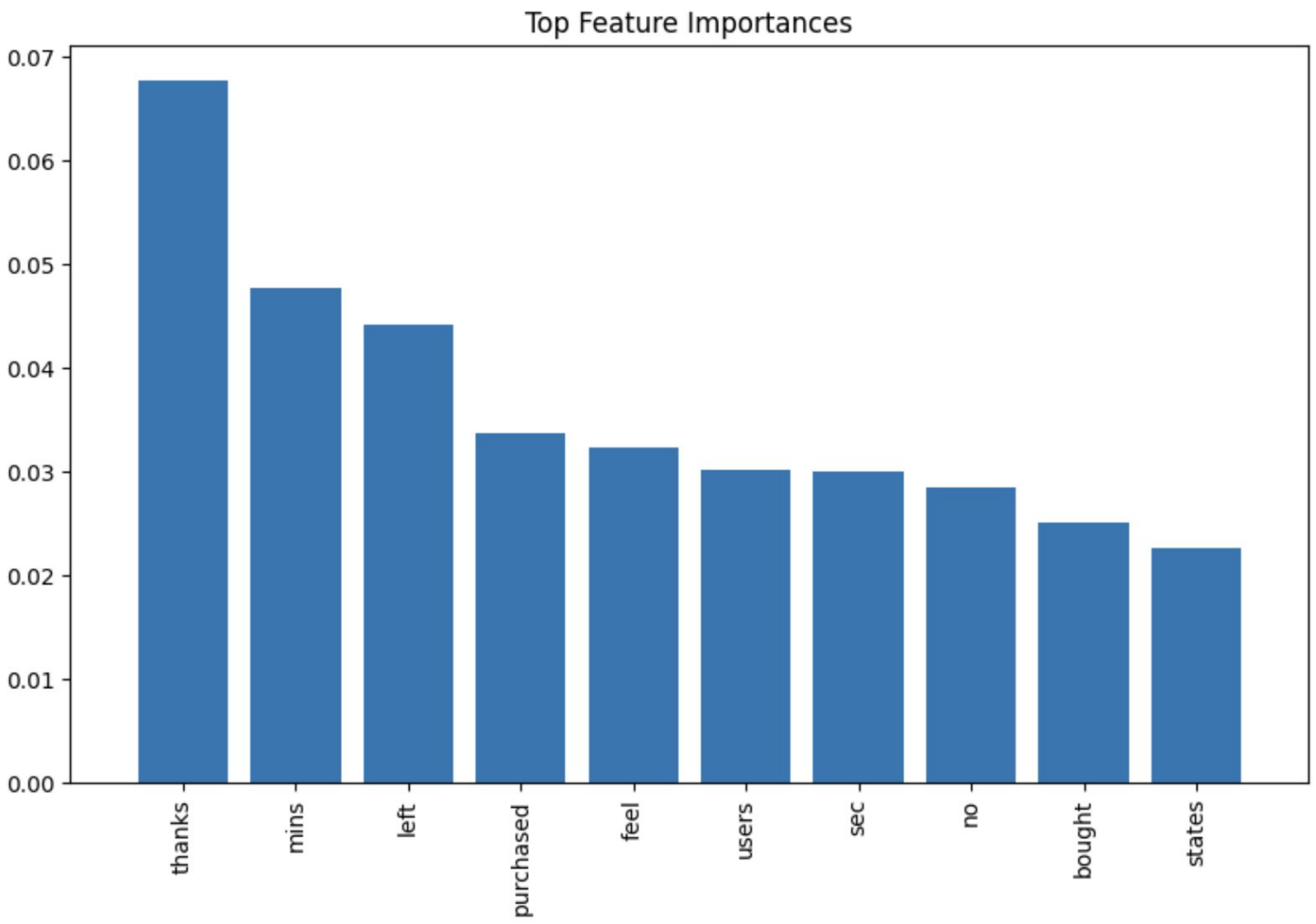
F1 Score: 0.9225

	precision	recall	f1-score	support
Misdirection	0.94	0.80	0.86	59
Not Dark Pattern	0.87	0.92	0.89	85
Scarcity	0.98	0.96	0.97	125
Social Proof	0.96	0.96	0.96	94
Urgency	0.84	0.92	0.88	63
accuracy			0.92	426
macro avg	0.92	0.91	0.91	426
weighted avg	0.92	0.92	0.92	426


Confusion Matrix
True Label
Predicted Label
Misdirection
Not Dark Pattern
Scarcity
Social Proof
Urgency
47
4
0
2
6
2
78
1
2
2
0
4
120
0
1
0
2
0
90
2
1
2
2
0
58
120
100
80
60
40
20
0



Learning Curve
Score
Training examples
1.00
0.95
0.90
0.85
0.80
0.75
100
200
300
400
500
600
700
800
Training score
Cross-validation score


Top Feature Importances
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
thanks
mins
left
purchased
feel
users
sec
no
bought
states

Future Steps:

- other feature extraction techniques (e.g. n-grams)
- OvA classification to gain insight into feature importance
- custom scoring function– higher penalty to non-dark patterns labelled as dark patterns