

# Predicting Game Result

Presented by:

Santosh Suwal  
HaoNan Ou  
Zhiqiang Wang

# What is League of Legends (LOL)?

- Multiplayer online battle arena
- 2 teams: **Blue** and **Red**
- Goal: To take down the enemy Nexus to win the game



# Data Overview

## First 10 minutes of LOL diamond ranked games

- 40 columns & 9879 rows
- Goal: Use a classification model to predict the outcome of a game based on the first 10 minutes statistics

# Data Exploration & Data Cleaning

# Data Exploration

- Game ID: To indicate each game is unique
- 19 features from each team (total 38) are collected after 10 minutes in game

redWardsPlaced  
redWardsDestroyed  
redFirstBlood  
redKills  
redDeaths  
redAssists  
redEliteMonsters  
redDragons  
redHeralds  
redTowersDestroyed  
redTotalGold  
redAvgLevel  
redTotalExperience  
redTotalMinionsKilled  
redTotalJungleMinionsKilled  
redGoldDiff  
redExperienceDiff  
redCSPerMin  
redGoldPerMin

blueWardsPlaced  
blueWardsDestroyed  
blueFirstBlood  
blueKills  
blueDeaths  
blueAssists  
blueEliteMonsters  
blueDragons  
blueHeralds  
blueTowersDestroyed  
blueTotalGold  
blueAvgLevel  
blueTotalExperience  
blueTotalMinionsKilled  
blueTotalJungleMinionsKilled  
blueGoldDiff  
blueExperienceDiff  
blueCSPerMin  
blueGoldPerMin

# Target Variable

Target Value: **blueWins**

**1 = Blue team wins**

**0 = Red team wins**

```
df['blueWins'].value_counts()
```

```
0      4949
```

```
1      4930
```

```
Name: blueWins, dtype: int64
```

# Correlation between Features

```
columns = df.columns
for i in columns:
    for j in columns:
        if abs(df[i].corr(df[j])) >= 0.95:
            if i != j:
                print([i, j, df[i].corr(df[j])])
                columns = columns.drop(i)
```

```
['blueFirstBlood', 'redFirstBlood', -1.0]
['blueKills', 'redDeaths', 1.0]
['blueDeaths', 'redKills', 1.0]
['blueTotalGold', 'blueGoldPerMin', 1.0]
['blueTotalMinionsKilled', 'blueCSPerMin', 1.0]
['blueGoldDiff', 'redGoldDiff', -1.0]
['blueExperienceDiff', 'redExperienceDiff', -1.0]
['redTotalGold', 'redGoldPerMin', 1.0]
['redTotalMinionsKilled', 'redCSPerMin', 0.9999999999999996]
```



# Data Cleaning

We dropped the following columns:

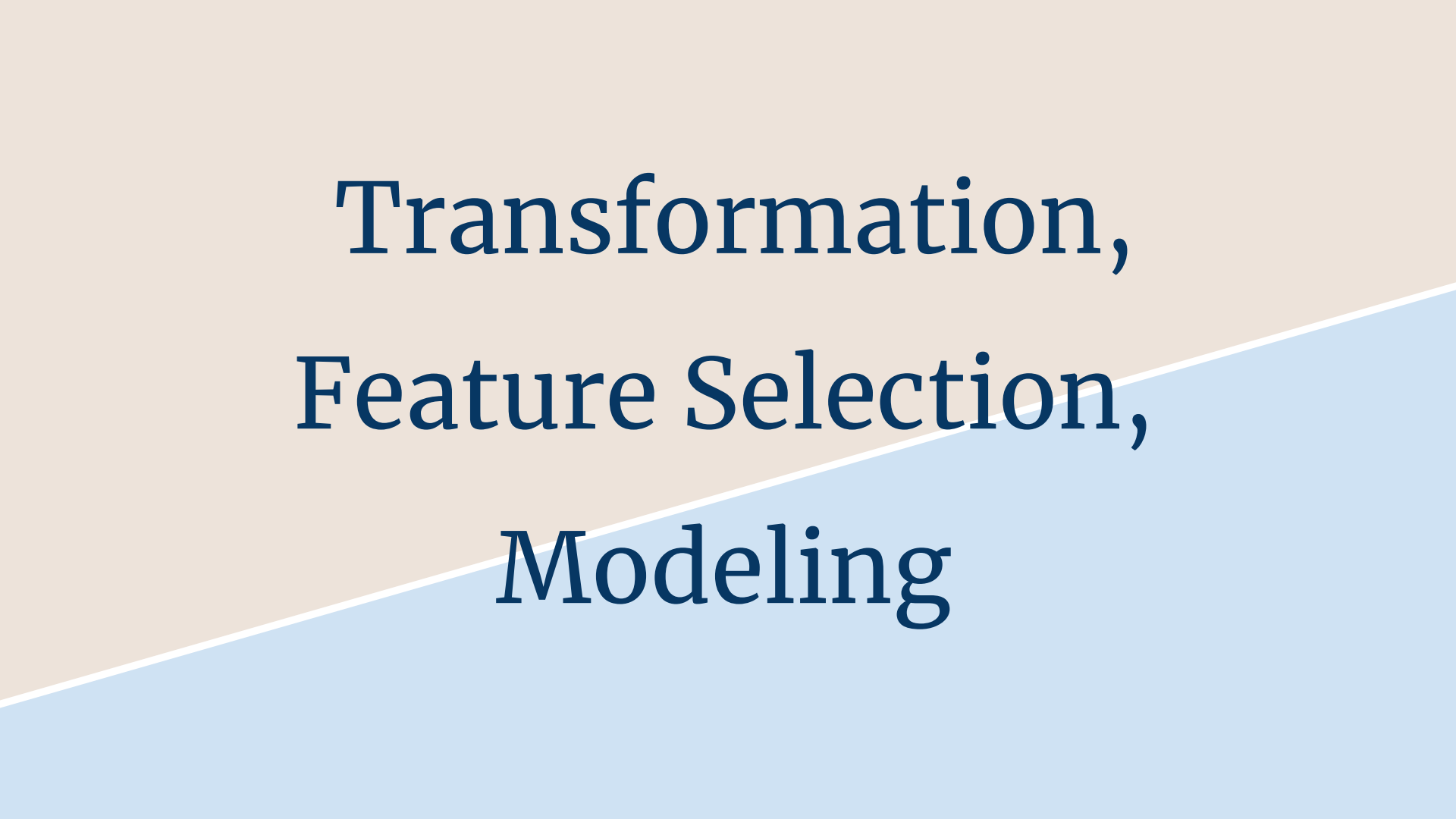
- Inversely correlated
- Perfectly correlated
- All derived values

# Finalized Data

RangeIndex: 9879 entries, 0 to 9878

Data columns (total 26 columns):

#	Column	Non-Null Count	Dtype
0	blueWins	9879 non-null	int64
1	blueWardsPlaced	9879 non-null	int64
2	blueWardsDestroyed	9879 non-null	int64
3	blueFirstBlood	9879 non-null	int64
4	blueKills	9879 non-null	int64
5	blueDeaths	9879 non-null	int64
6	blueAssists	9879 non-null	int64
7	blueDragons	9879 non-null	int64
8	blueHeralds	9879 non-null	int64
9	blueTowersDestroyed	9879 non-null	int64
10	blueTotalGold	9879 non-null	int64
11	blueAvgLevel	9879 non-null	float64
12	blueTotalExperience	9879 non-null	int64
13	blueTotalMinionsKilled	9879 non-null	int64
14	blueTotalJungleMinionsKilled	9879 non-null	int64
15	redWardsPlaced	9879 non-null	int64
16	redWardsDestroyed	9879 non-null	int64
17	redAssists	9879 non-null	int64
18	redDragons	9879 non-null	int64
19	redHeralds	9879 non-null	int64
20	redTowersDestroyed	9879 non-null	int64
21	redTotalGold	9879 non-null	int64
22	redAvgLevel	9879 non-null	float64
23	redTotalExperience	9879 non-null	int64
24	redTotalMinionsKilled	9879 non-null	int64
25	redTotalJungleMinionsKilled	9879 non-null	int64



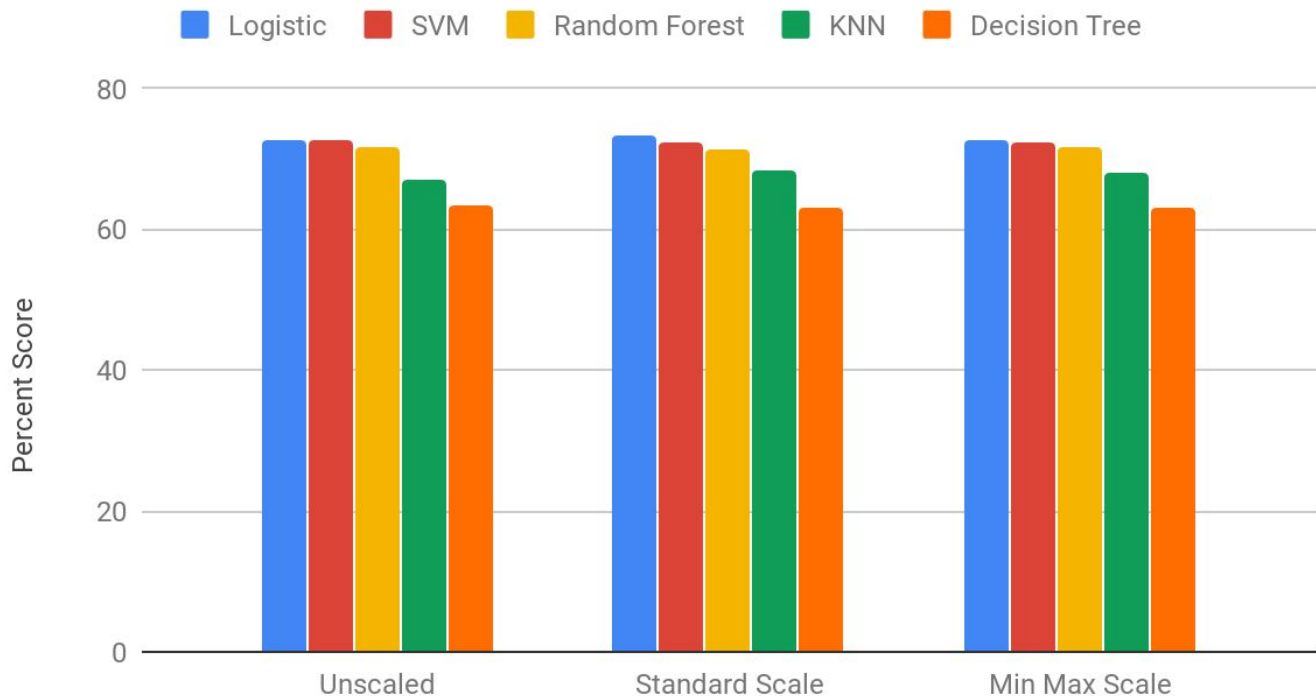
# Transformation, Feature Selection, Modeling

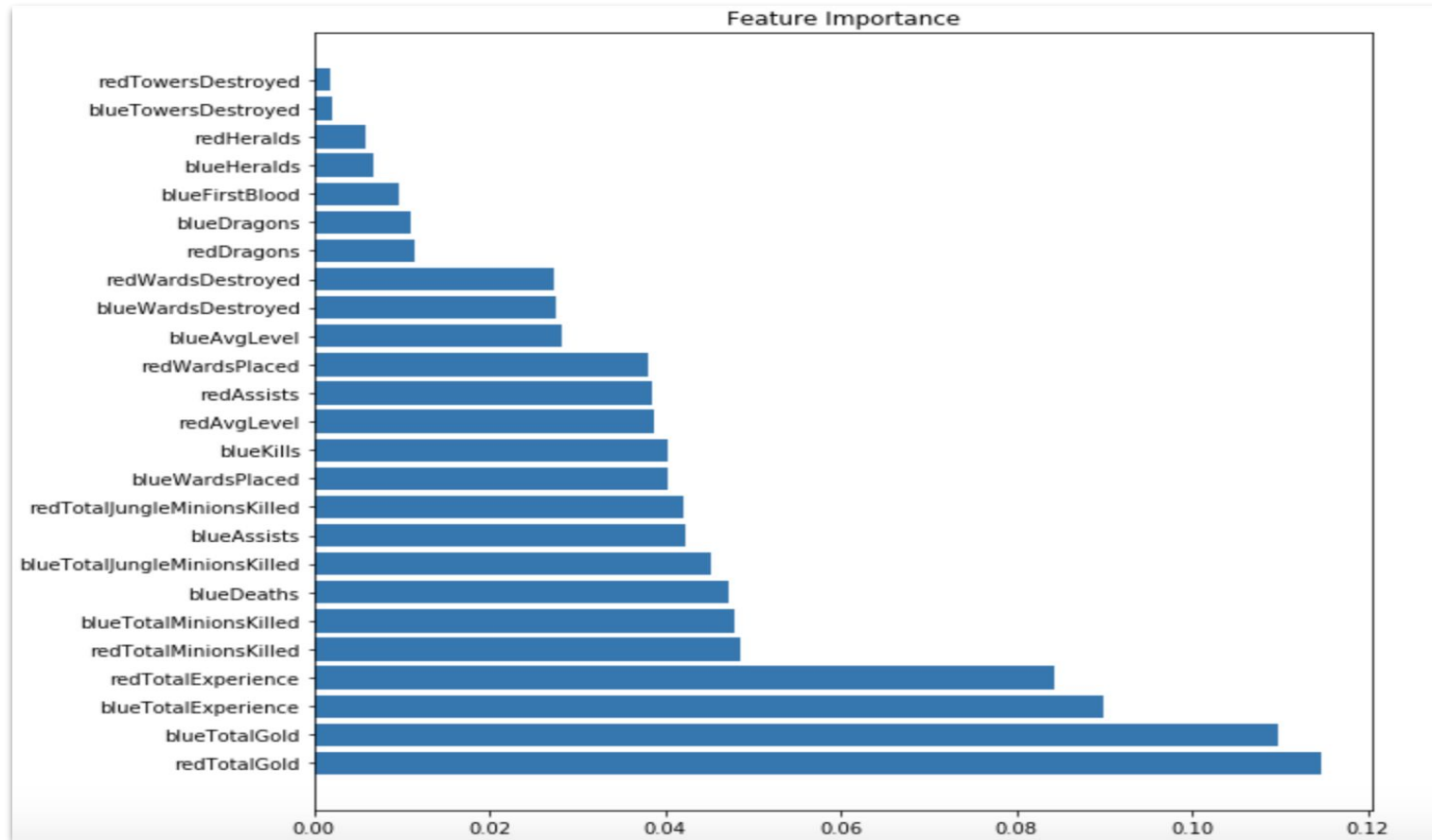
# Feature Scaling

- Normalization via min-max scaling
- Standardization

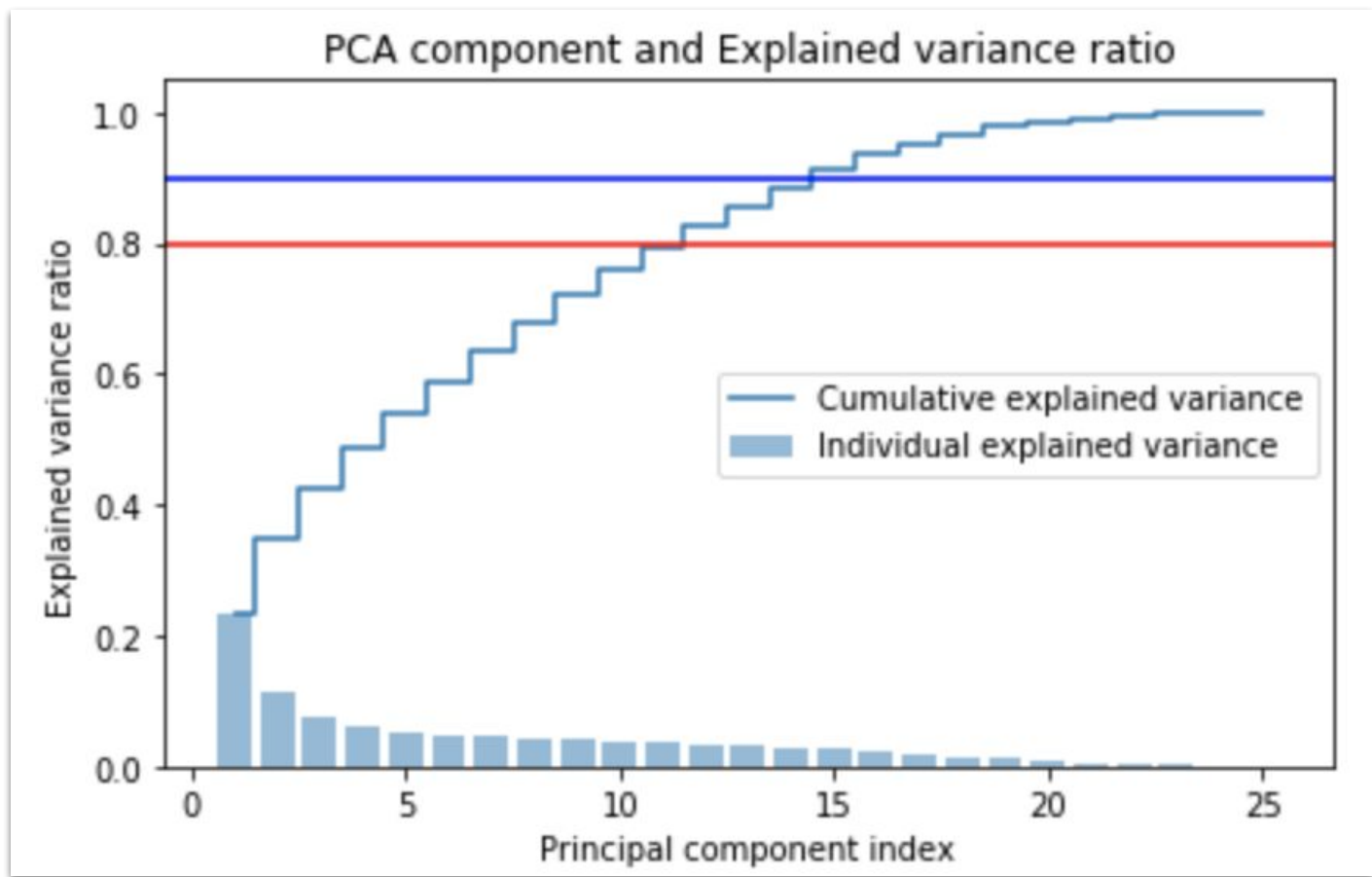
# Classification

Unscaled, Standard Scale and Min Max Scale

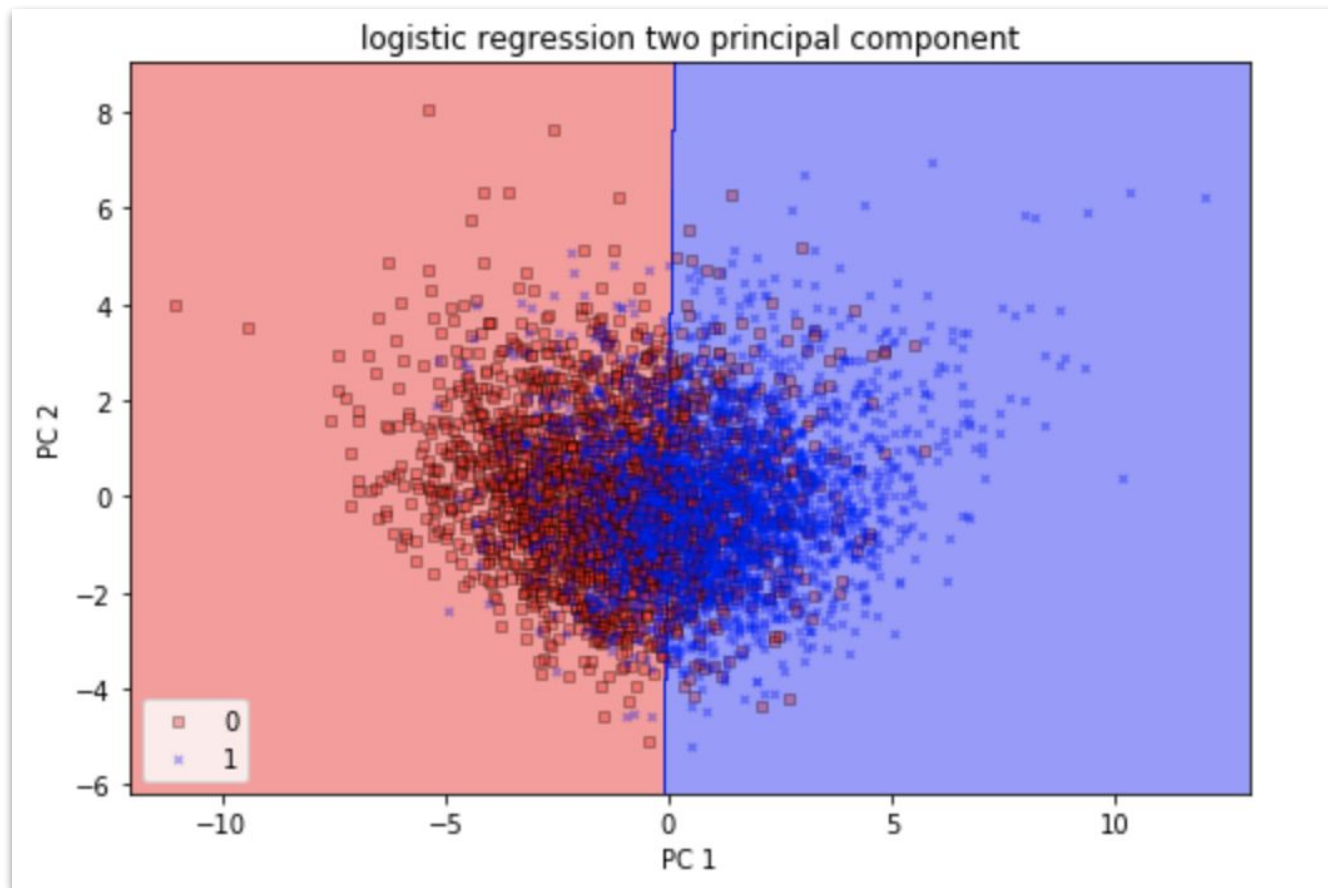




# Important Feature

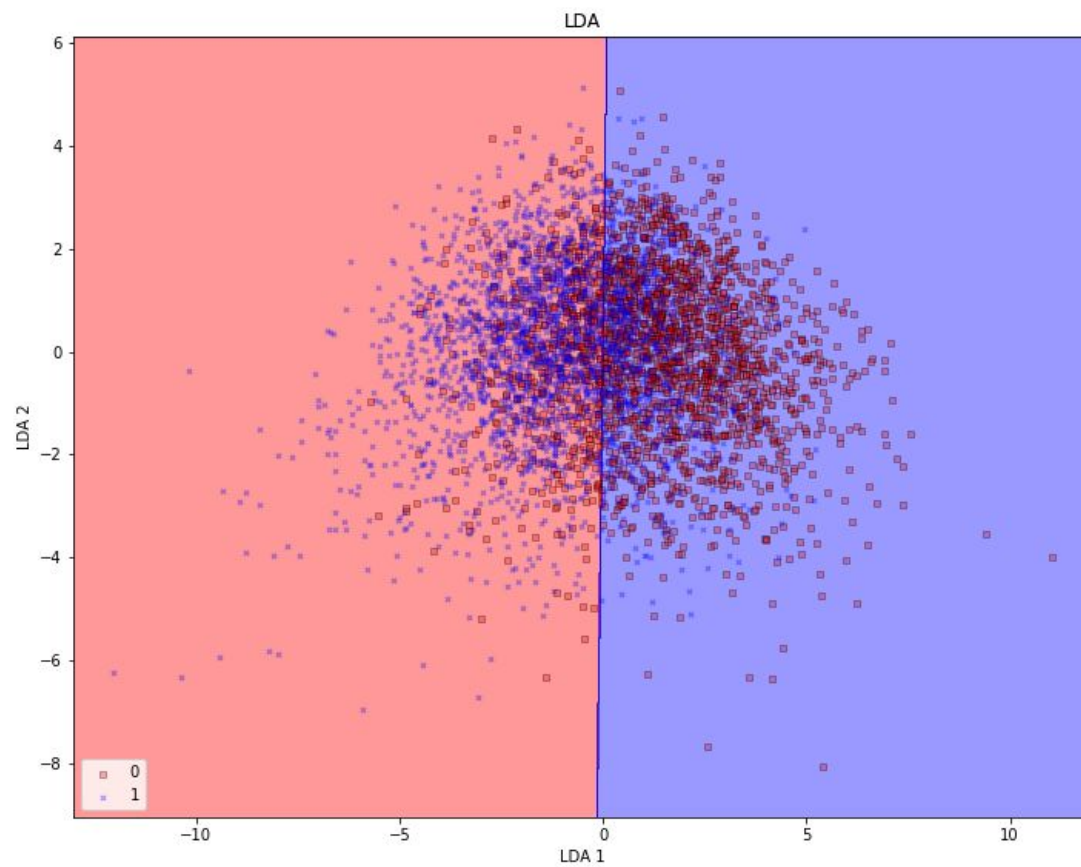


# PCA



# PCA Plot



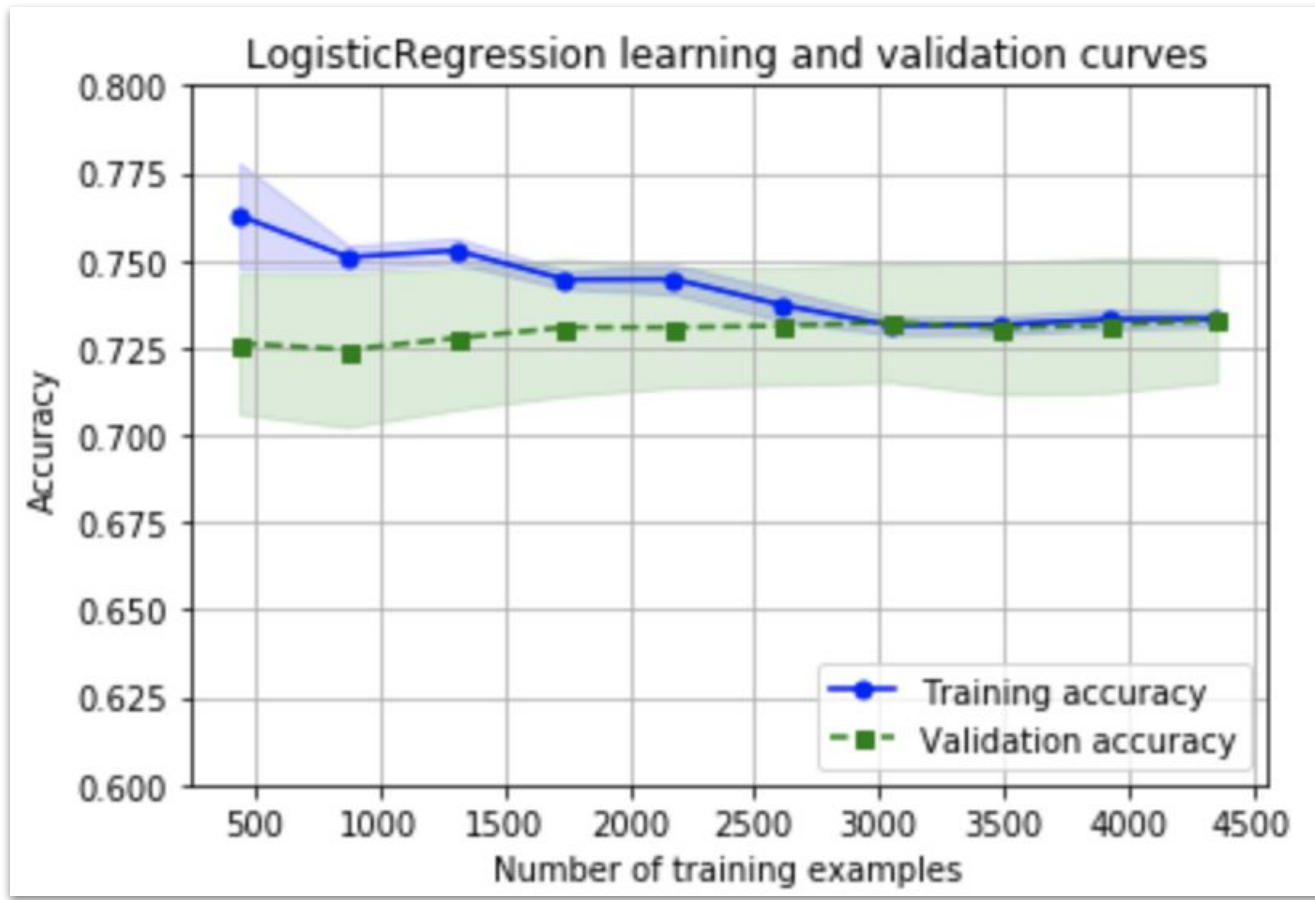


# LDA

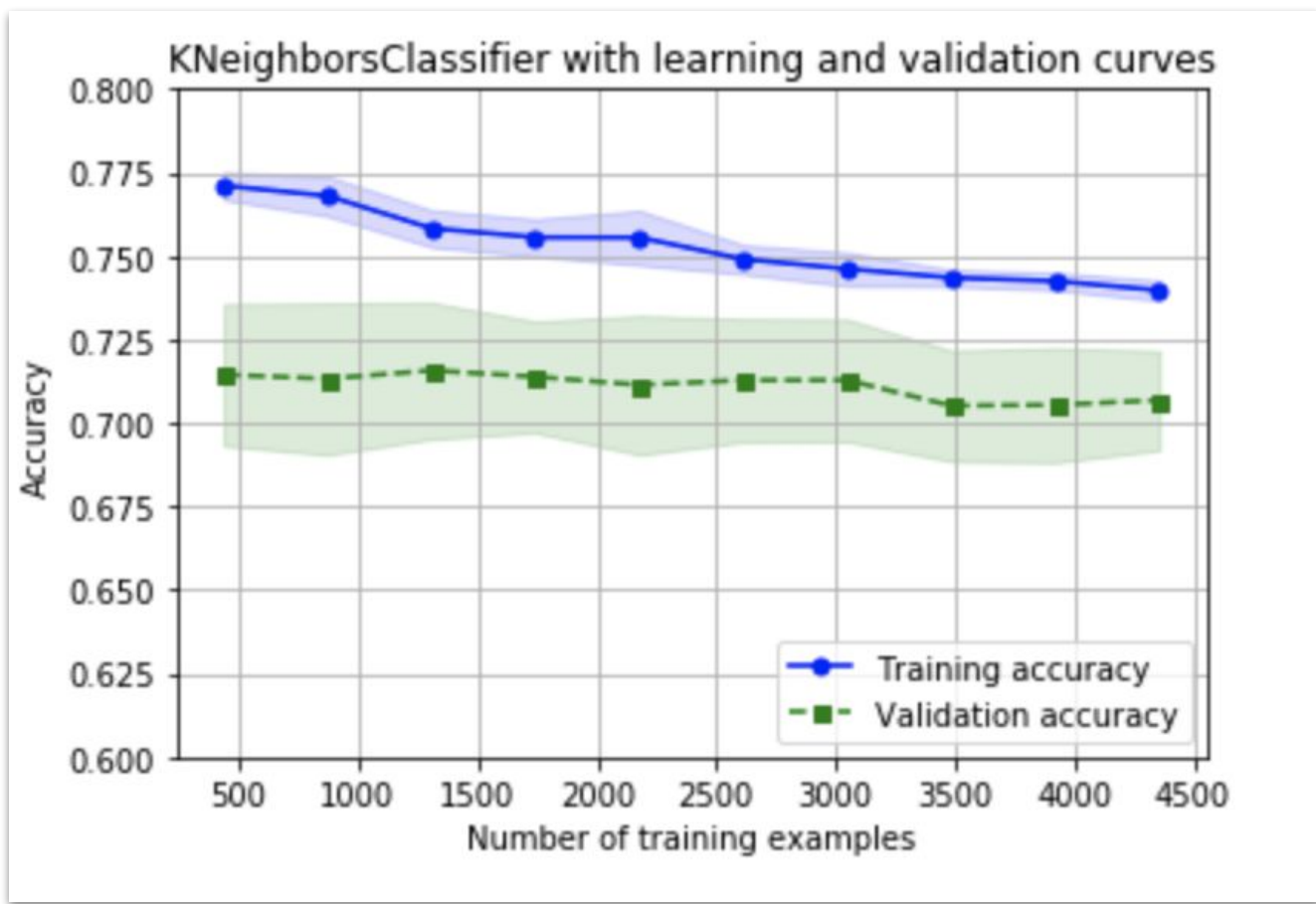
# Grid Search CV

Best Params	Score
<ul style="list-style-type: none"><li>● LogisticRegression<ul style="list-style-type: none"><li>○ C = 0.001</li></ul></li></ul>	72.7%
<ul style="list-style-type: none"><li>● KNeighborsClassifier<ul style="list-style-type: none"><li>○ n_neighbors = 21</li></ul></li></ul>	70.1%
<ul style="list-style-type: none"><li>● SVM<ul style="list-style-type: none"><li>○ C = 0.01, kernel='rbf'</li></ul></li></ul>	72.5%

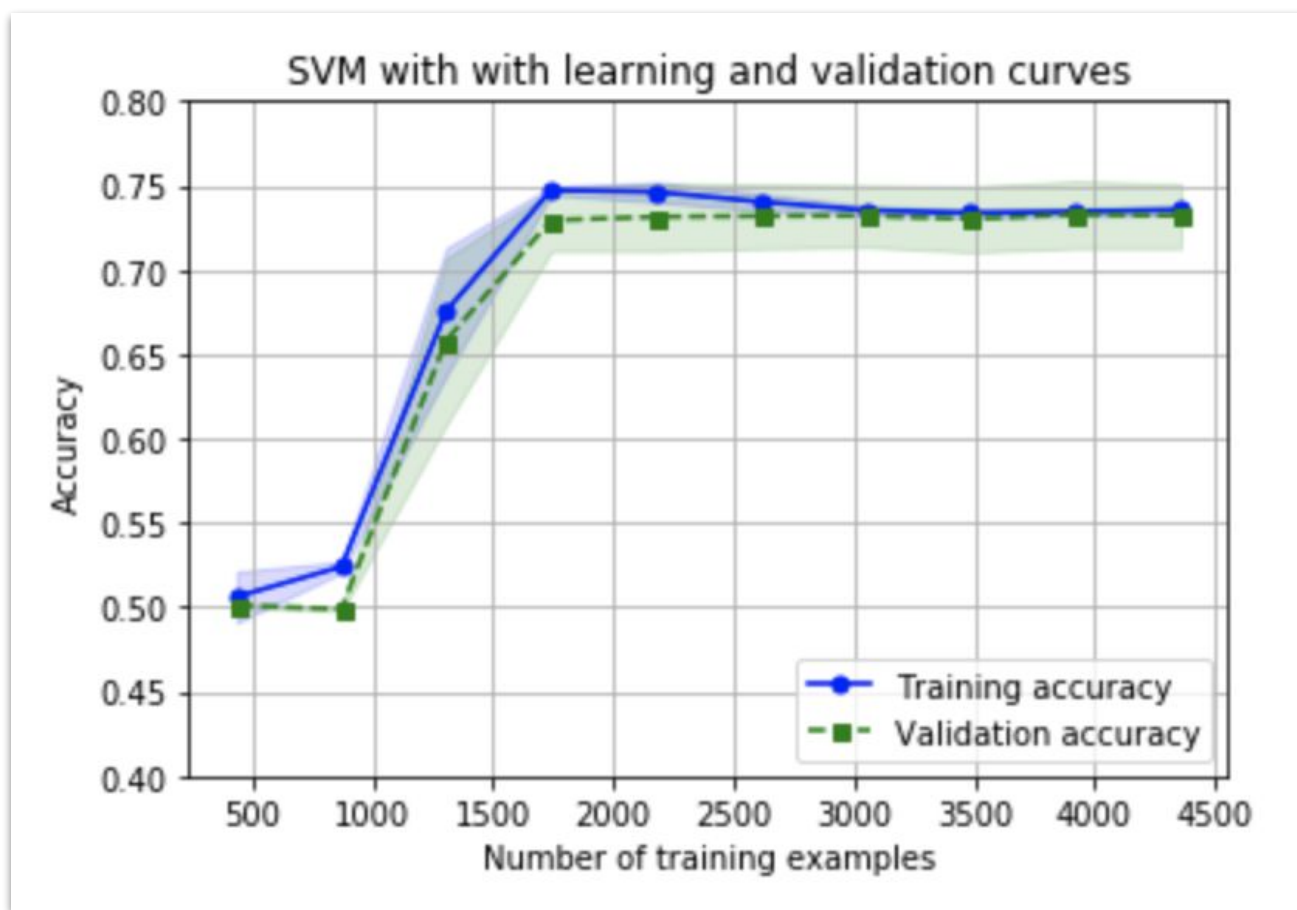
# Metrics, Validation, Evaluation



# Learning and Validation Curves



# Learning and Validation Curves

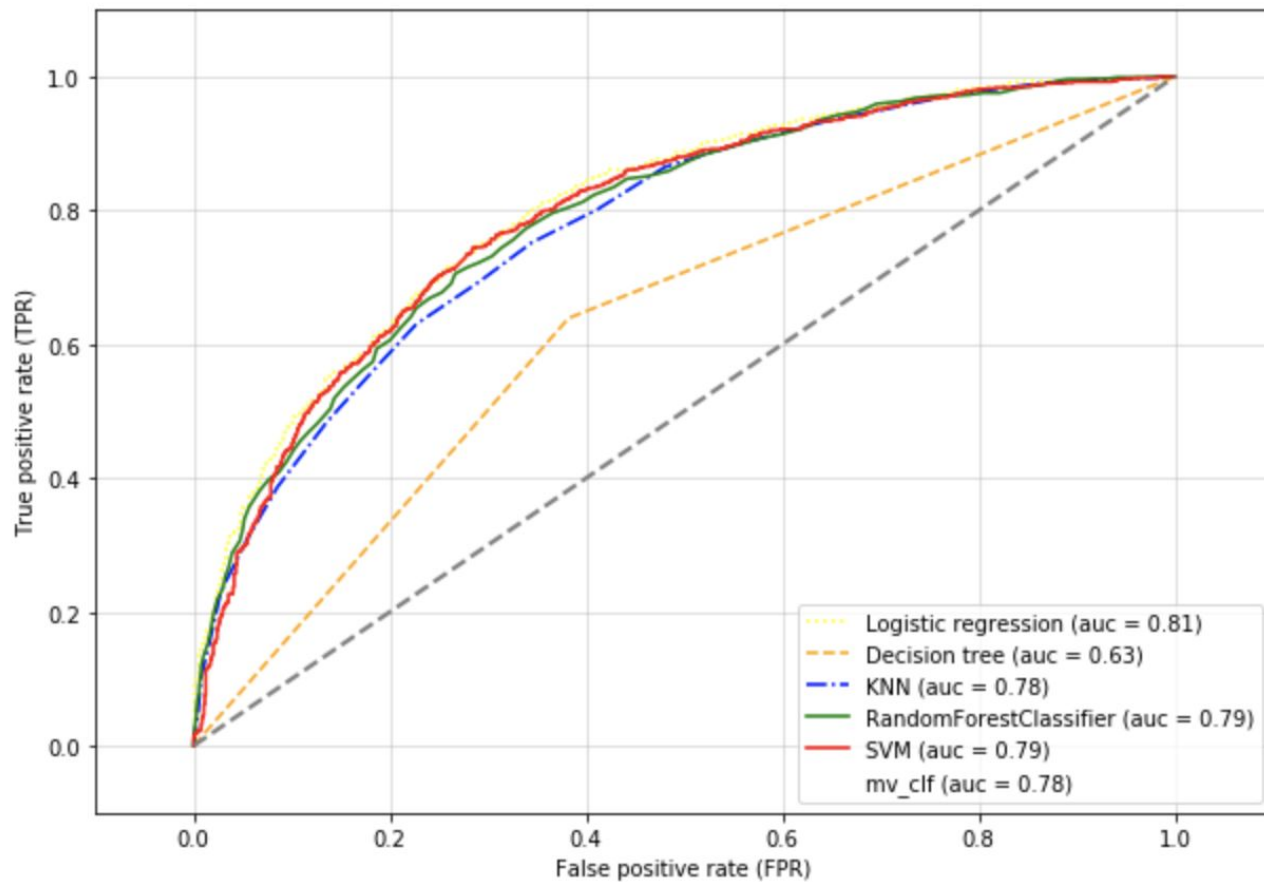


# Learning and Validation Curves

# Evaluating the Model Performance of Each Classifier

10-fold cross validation:

ROC AUC:	0.81	(+/- 0.02)	[Logistic regression]
ROC AUC:	0.63	(+/- 0.01)	[Decision tree]
ROC AUC:	0.78	(+/- 0.02)	[KNN]
ROC AUC:	0.79	(+/- 0.02)	[RandomForestClassifier]
ROC AUC:	0.79	(+/- 0.02)	[SVM]
ROC AUC:	0.78	(+/- 0.01)	[mv_clf]



# ROC Curve



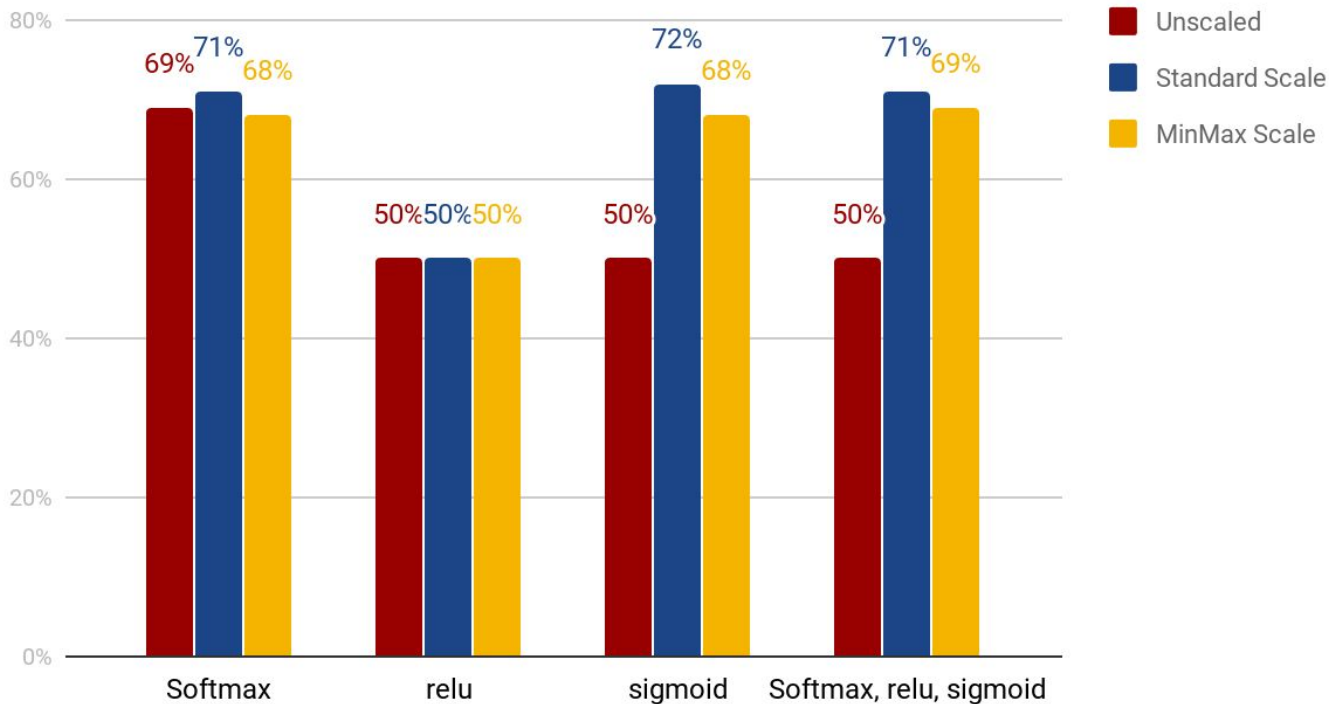
# Conclusion

## Accuracy Score Table

	<b>Logistic</b>	<b>SVM</b>	<b>RandomForest</b>	<b>KNN</b>	<b>DecisionTree</b>
train data	72.7%	72.5%	71.7%	70.1%	63.1%
test data	72.3%	72.1%	72.1%	70.6%	64.0%

# Keras (Dense Layer)

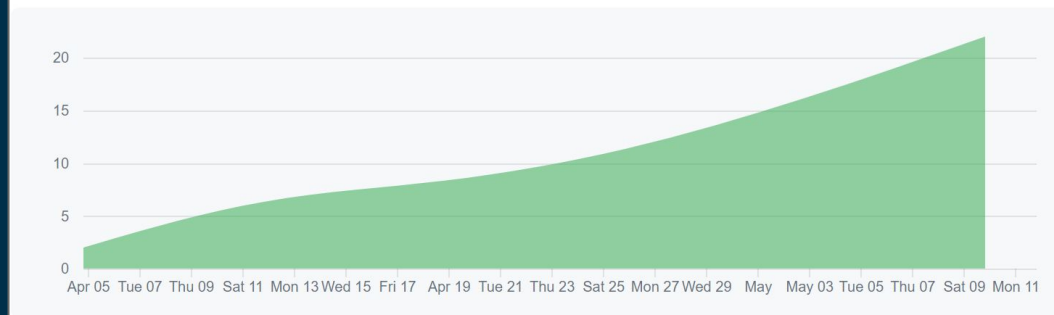
Score with Kares



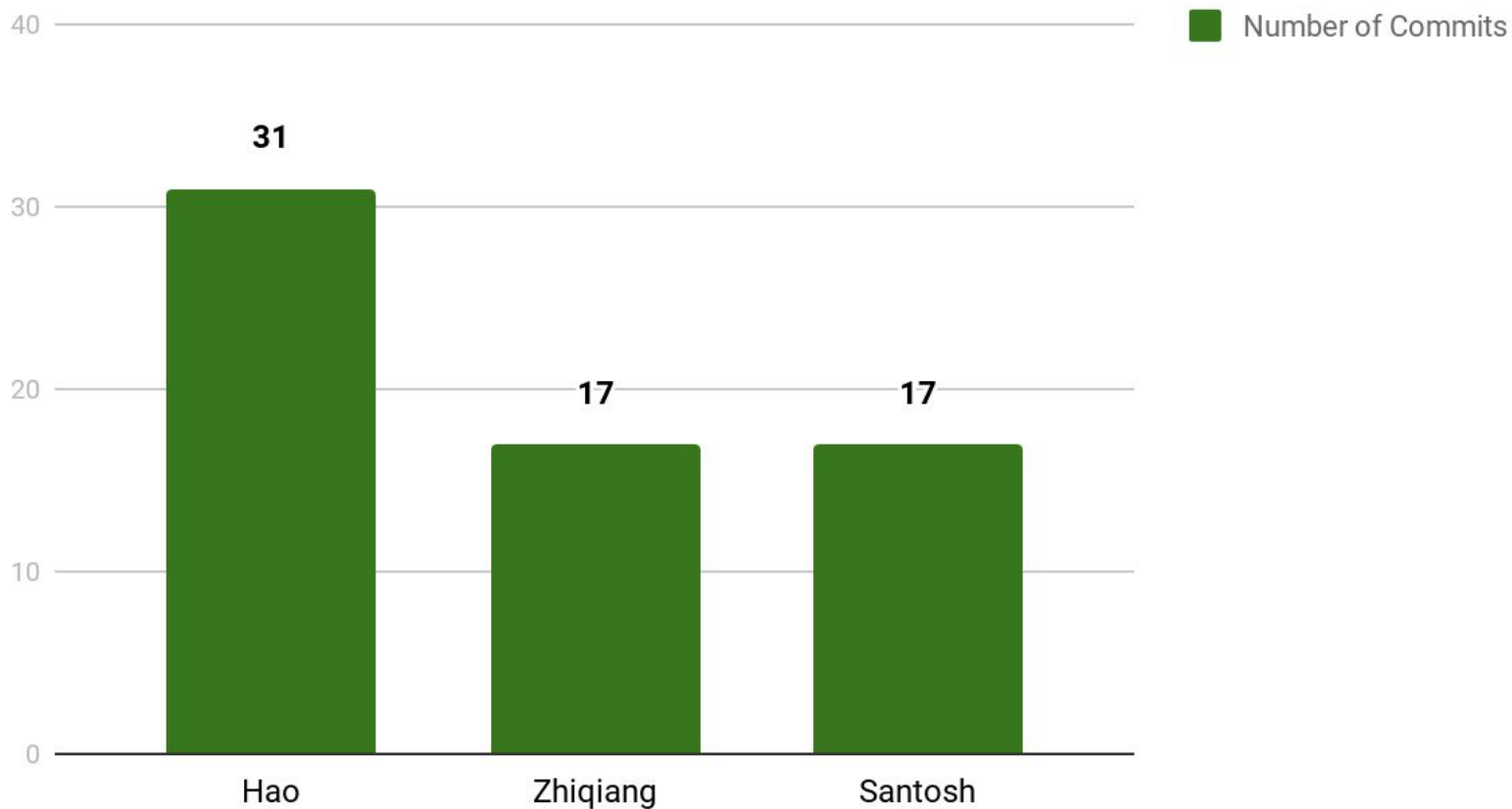
# Github

- 99 commits
- 30 Pull Requests
- 10 issues
- 5 meetings

Contributions to master, excluding merge commits



## Github Commits excluding merges and pull request



Thank You!

Q&A