European Football Match Predictor

October 30, 2018

Standard Chartered
AIA
LIVERPOOL WAY

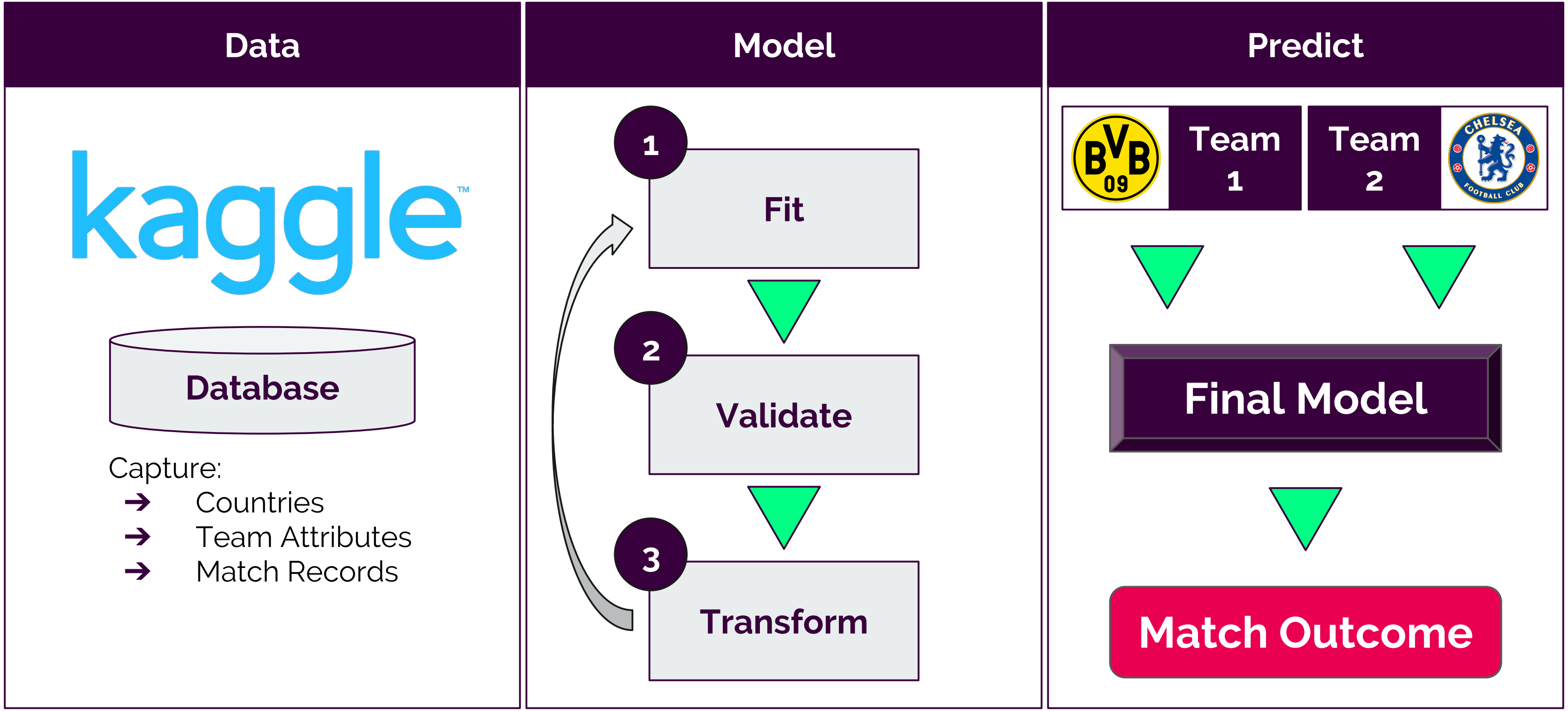
Goal

Predict the final outcome of a European football match

Team 1 WIN

Team 2

Approach



Data

kaggle

Database

Capture:

- ➔ Countries
- ➔ Team Attributes
- ➔ Match Records

Model

1 Fit

2 Validate

3 Transform

Predict

Team 1

Team 2

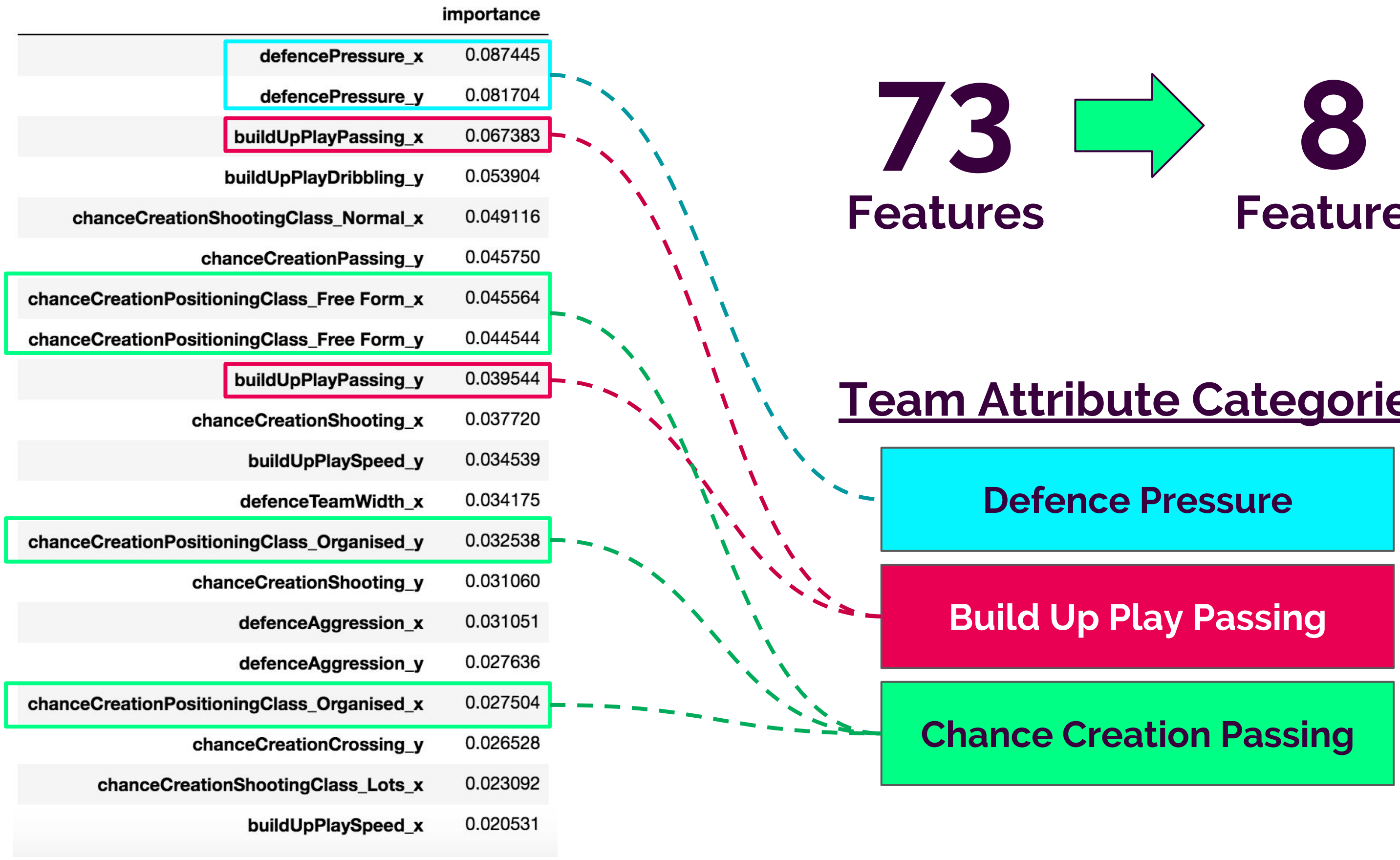
Final Model

Match Outcome

Tools

Tools	Purpose
SQLite	● Data Querying
scikit learn SM StatsModels Statistics in Python pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ NumPy	● Model ● Feature Engineering
matplotlib Seaborn HTML 5 CSS 3 Flask	● Data Visualization ● Web UI

Feature Importances

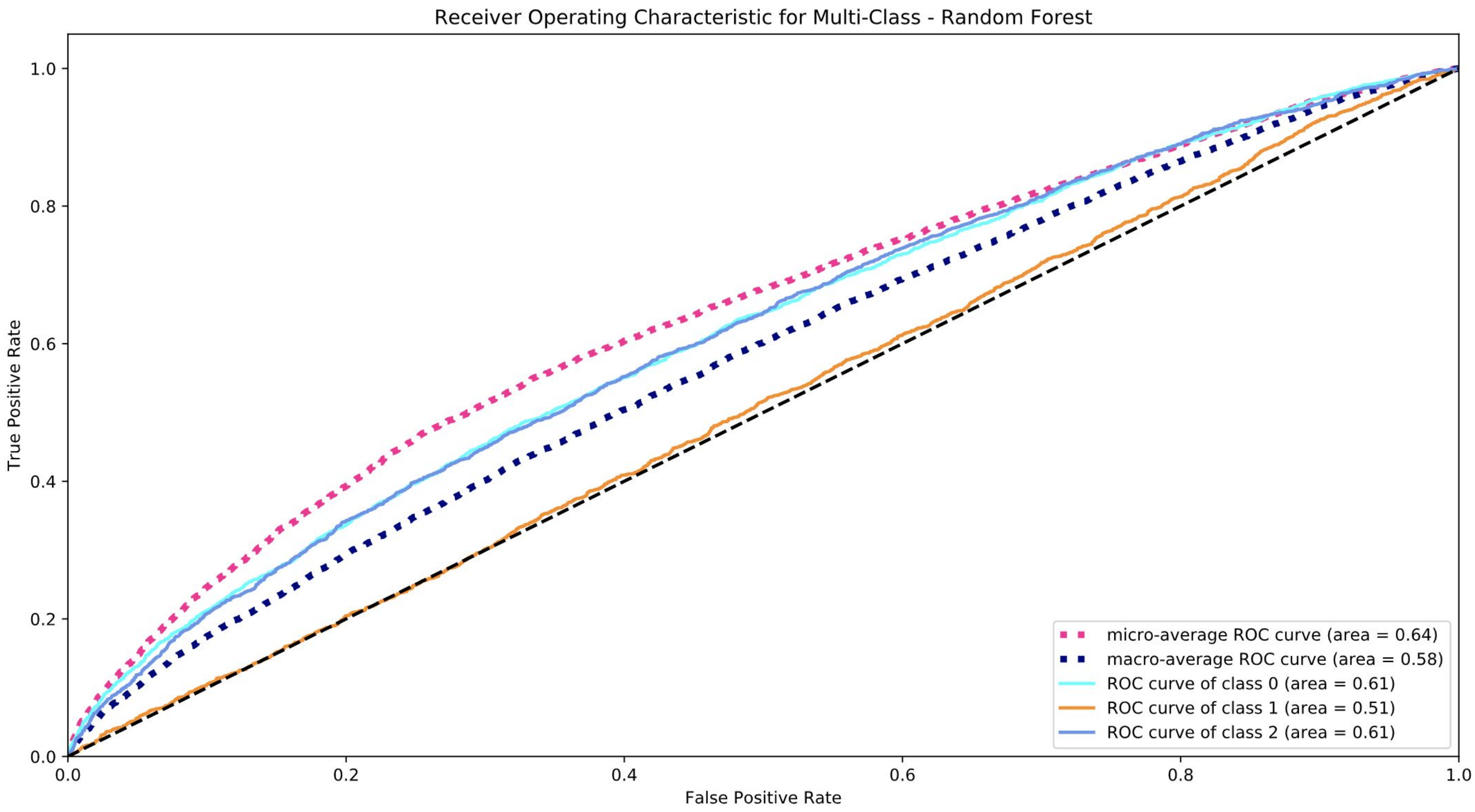


	importance
defencePressure_x	0.087445
defencePressure_y	0.081704
buildUpPlayPassing_x	0.067383
buildUpPlayDribbling_y	0.053904
chanceCreationShootingClass_Normal_x	0.049116
chanceCreationPassing_y	0.045750
chanceCreationPositioningClass_Free Form_x	0.045564
chanceCreationPositioningClass_Free Form_y	0.044544
buildUpPlayPassing_y	0.039544
chanceCreationShooting_x	0.037720
buildUpPlaySpeed_y	0.034539
defenceTeamWidth_x	0.034175
chanceCreationPositioningClass_Organised_y	0.032538
chanceCreationShooting_y	0.031060
defenceAggression_x	0.031051
defenceAggression_y	0.027636
chanceCreationPositioningClass_Organised_x	0.027504
chanceCreationCrossing_y	0.026528
chanceCreationShootingClass_Lots_x	0.023092
buildUpPlaySpeed_x	0.020531

73 Features → **8 Features**

Team Attribute Categories

ROC Curve Analysis

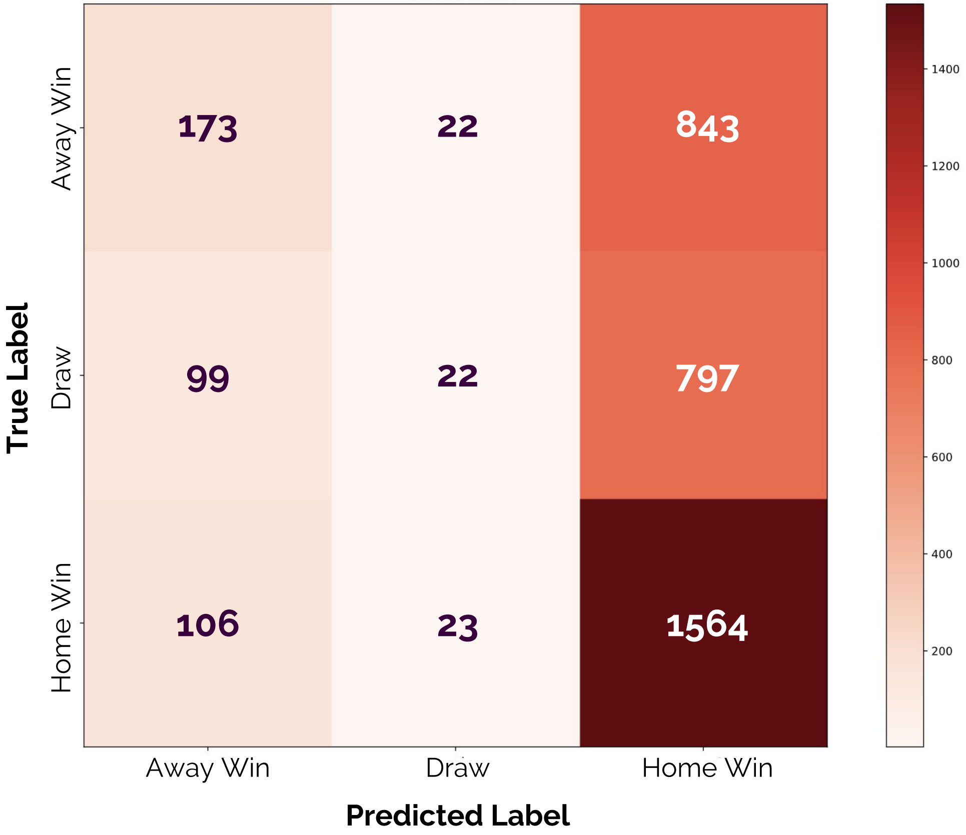


Area Under Curve (AUC) Comparison Matrix

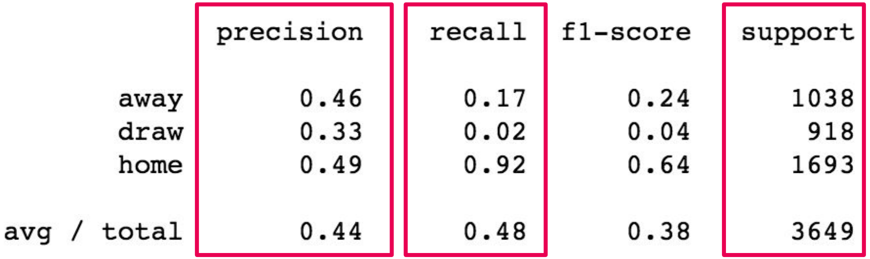
Model	Class ‘home win’ AUC	Class ‘draw’ AUC	Class ‘away win’ AUC	Micro-average AUC	Macro-average AUC
Random Forest	**0.61**	**0.52**	**0.61**	**0.64**	**0.58**
SVM (rbf kernel)	0.54	0.50	0.54	0.59	0.53
Logistic Regression	0.59	**0.52**	0.59	0.63	0.56

* Higher AUC is better

Confusion Matrix Analysis



Random Forest Confusion Matrix



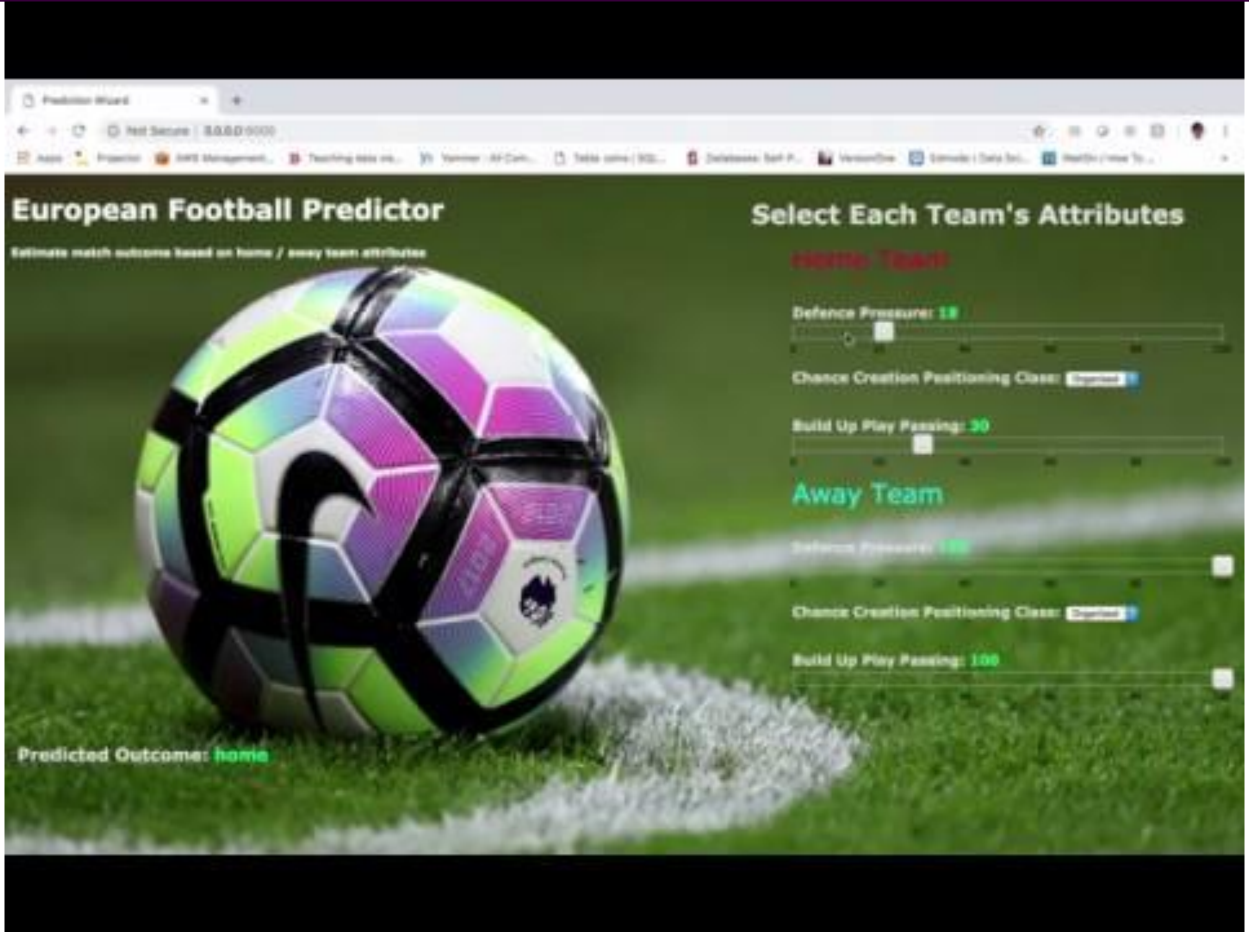
```
             precision    recall  f1-score   support

       away       0.46      0.17      0.24      1038
       draw       0.33      0.02      0.04       918
       home       0.49      0.92      0.64      1693

avg / total       0.44      0.48      0.38      3649
```

Model predicts the true outcome less than 50%

Model frequently predicts 'Home Win'

Class Imbalance

Web User Interface



European Football Predictor

Estimate match outcome based on home / away team attributes

Predicted Outcome: home

Select Each Team's Attributes

Home Team

Defence Pressure: 18

Chance Creation Positioning Class: Organised

Build Up Play Passing: 30

Away Team

Defence Pressure: 100

Chance Creation Positioning Class: Organised

Build Up Play Passing: 100

Conclusion

- Class Imbalance is adversely affecting the current model

- Additional exploratory data analysis required

Recommendation

Stick with random guessing until further notice...

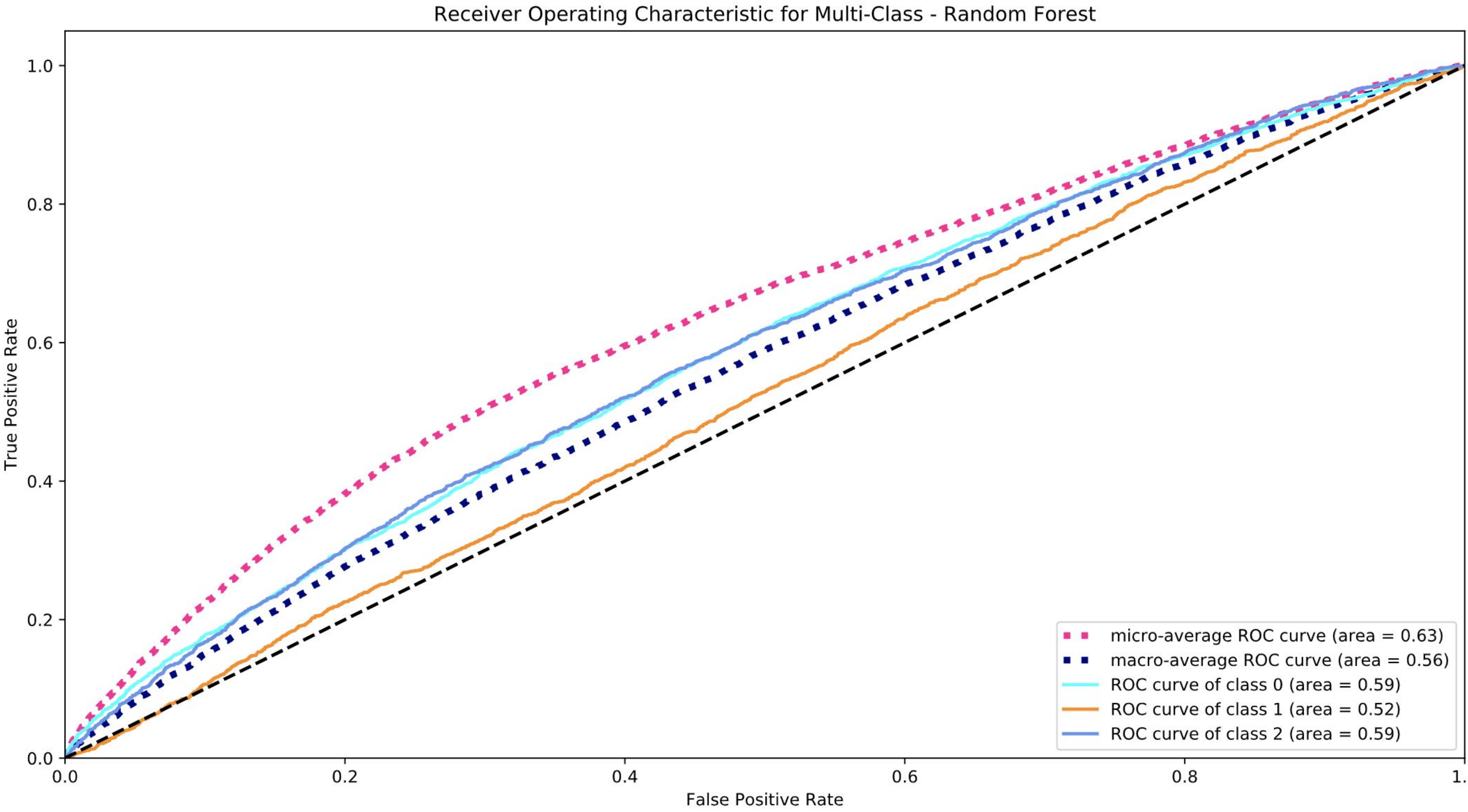
Next Steps

- SMOTE
- More Data
- Model Enhancements
- Data Visualization Enhancements

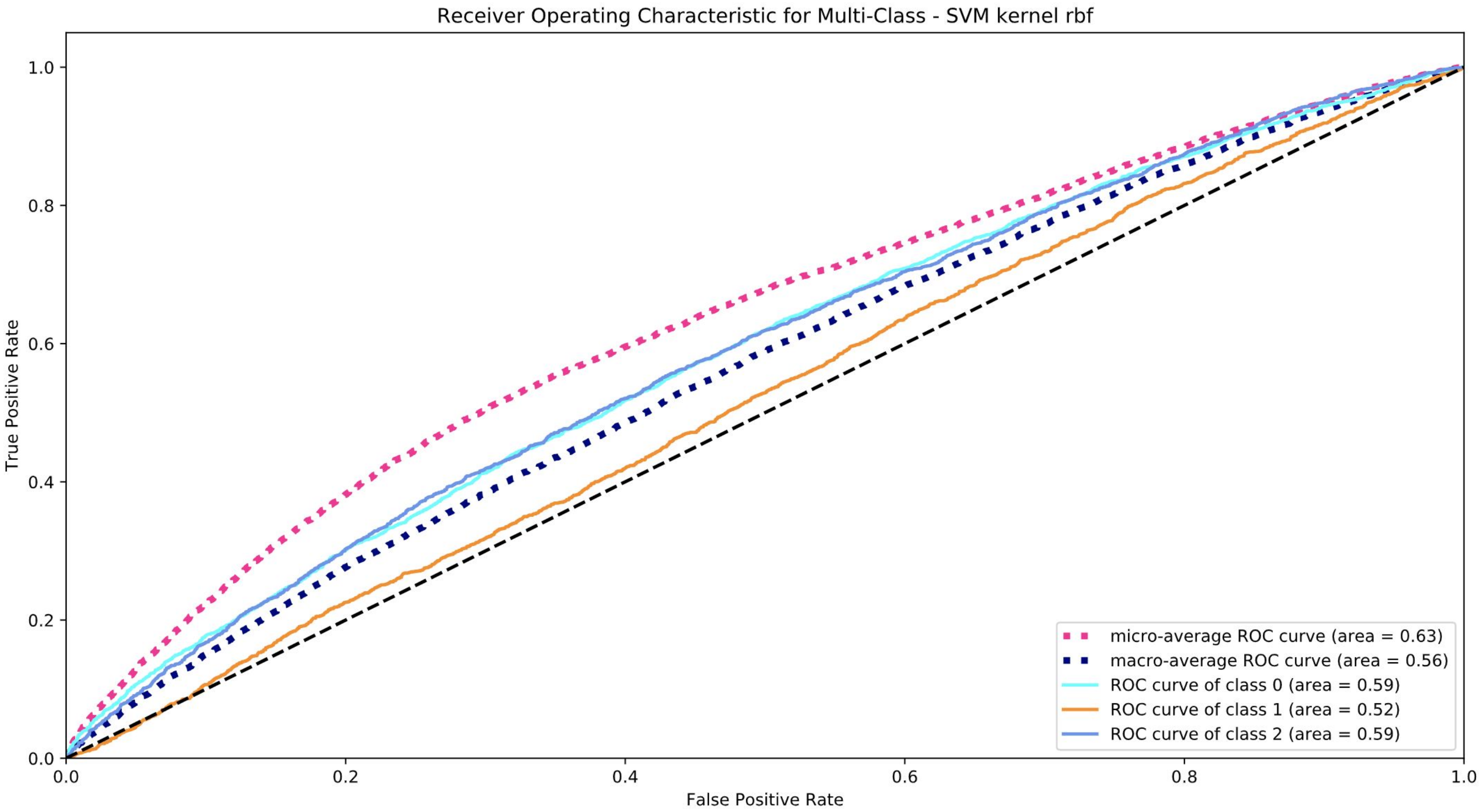
Questions?

Appendix

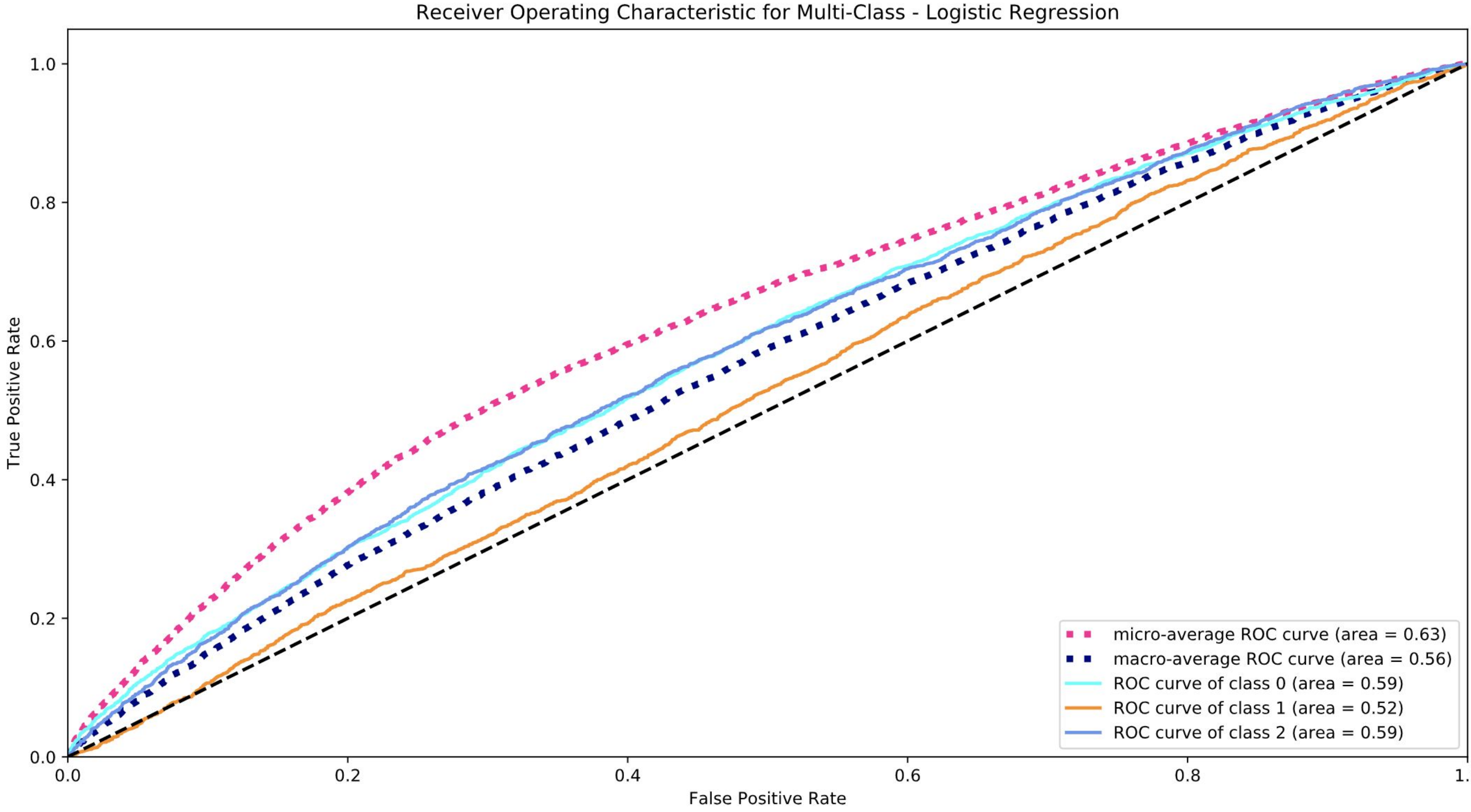
ROC Curve - Random Forest



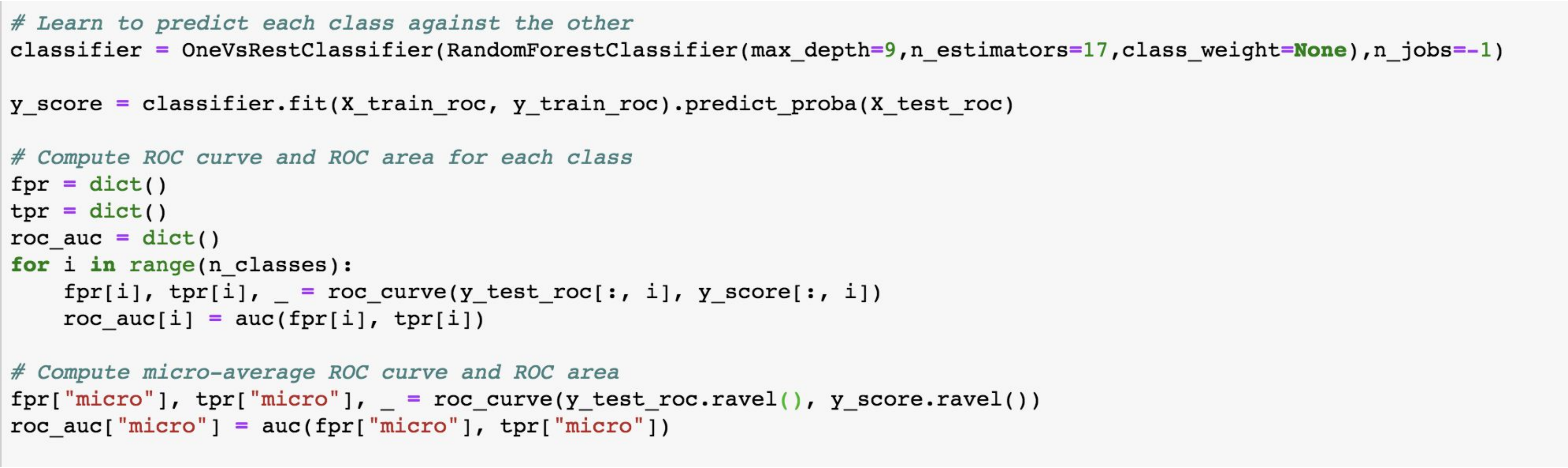
ROC Curve - SVM rbf



ROC Curve - Logistic Regression



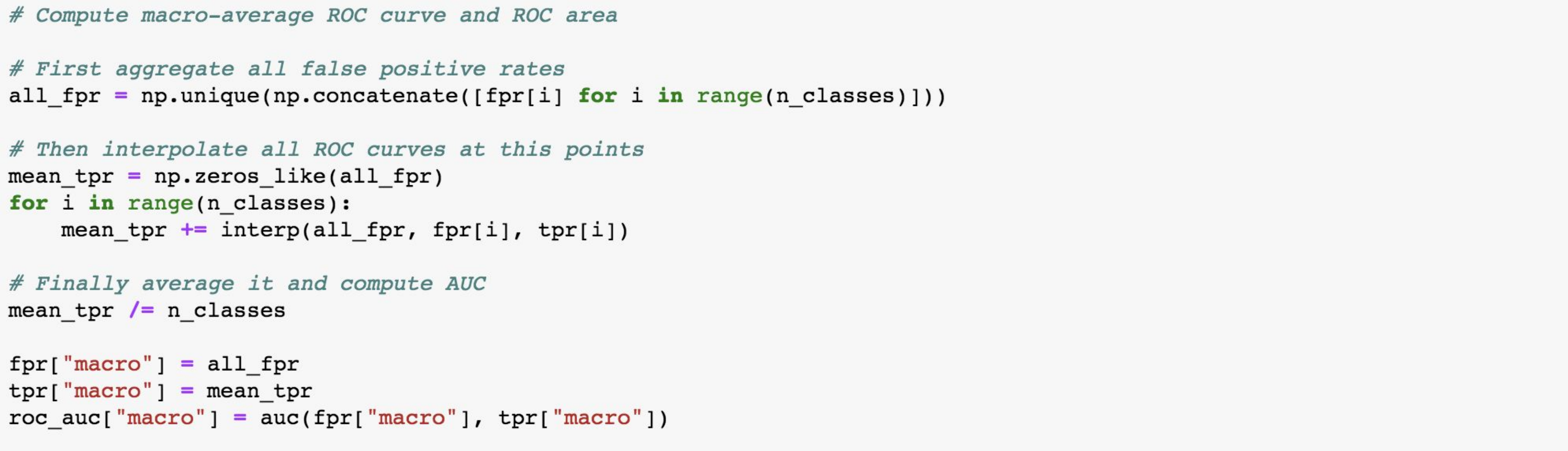
Calculating ROC Curves - code part 1



```
# Learn to predict each class against the other
classifier = OneVsRestClassifier(RandomForestClassifier(max_depth=9,n_estimators=17,class_weight=None),n_jobs=-1)

y_score = classifier.fit(X_train_roc, y_train_roc).predict_proba(X_test_roc)

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_roc[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test_roc.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

SKLearn Reciver Operating Characteristic (ROC) for multi-class

Calculating ROC Curves - code part 2



```
# Compute macro-average ROC curve and ROC area

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
```

SKLearn Reciver Operating Characteristic (ROC) for multi-class