**Instructions on Reproducing the Predictions**

## Data pre-processing

1. Load libraries
   * Pandas
   * Numpy
   * Matplotlib.pyplot
   * Seaborn
   * Sklearn
   * Random
   * sklearn.svm import SVC
   * sklearn.tree import DecisionTreeClassifier, plot\_tree
   * sklearn.model\_selection import train\_test\_split
   * sklearn import metrics
   * sklearn import tree
   * sklearn.linear\_model import LogisticRegression
   * sklearn.linear\_model import LogisticRegressionCV
   * sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay
   * sklearn.model\_selection import GridSearchCV
   * sklearn.decomposition import PCA
   * sklearn.metrics import balanced\_accuracy\_score
   * imblearn.over\_sampling import SMOTE
2. Load data from Github:
   * Pbp.csv : <https://raw.githubusercontent.com/samhoppen/DSC-540-Group-Project/main/SIS%20PBP.csv>
   * Game\_info.csv: <https://raw.githubusercontent.com/samhoppen/DSC-540-Group-Project/main/SIS%20Game%20Info.csv>
   * Player\_positions.csv: <https://raw.githubusercontent.com/samhoppen/DSC-540-Group-Project/main/SIS%20Player%20Positions.csv>
3. Simplify the route naming by replacing 'Route' in *Player-positions.csv* file into Curl, Out, Slant, Dig, Drag, Post, Flat, Screen, Corner, Swing, Comeback, Angle and Other based on the route description.
4. Merge *Pbp.csv* and *modified route Player\_position.csv* by its 'GameID', 'EventID' into a new dataframe
5. Convert and count the number of occurrences for each 'Route' type for each play and convert the counts into columns for a new dataframe.
6. Merge the new dataframe created in step 4 into the merged *Pbp.csv* and *modified route Player\_position.csv* dataframe.
7. Simplify the drop type naming similar to renaming 'Route' process. Renaming it to Rollout, RPO, Stepback, Trick, Spike, and Other.
8. Create Success Column based on First Down and TouchDown where if either is 1, then the new column is 1 otherwise 0

## Feature Extraction

1. Use .describe() on the processed dataframe for the data summary
2. Create a bar chart of the pre-processed dataset’s Route distribution by x = "Route", data = player\_positions, order = player\_positions['Route'].value\_counts().index
3. Recreate a bar chart similar to step 2 with processed dataset x = "Route", data = valid\_routes, order = valid\_routes['Route'].value\_counts().index
4. Extract all numeric variables from the processed dataset into a new dataframe.
5. Create a correlation variable corr = numeric variables dataframe.corr()
6. Create a heatmap using the correlation variable.
7. Create a bar chart of the distribution of drop type group by 'success' using the processed dataset.

## Model Training and Prediction Generation

### Expected Success Model with Gradient Boosting

* From the main data, drop the following columns: 'GameID\_x', 'GameID\_y', 'EventID\_x', 'EventID\_y', 'Season', 'Week', 'PlayDesc', 'OffensiveTeam', 'DefensiveTeam', 'OffTeamScoreBefore', 'DefTeamScoreBefore', 'Quarter', 'TimeLeft', 'EventType', 'SideOfField', 'StartYard', 'KEY', 'FumbleByRusher', 'DropType', 'EPA'
* Set aside the success variable for the target
* Standardize the inputs using StandardScaler
* Create logistic regression model with a cross validation of 10 and fit using standardized data
* Display coefficients and evaluate variables with importance
* Create Gradient boost model and fit again using the standardized data
* Use the Gradient boost to predict probability of success
* Isolate the predictions, routes, original success, and defensive coverages to combine the data into a table for analysis

### Success based on Dropback Type

* Drop all columns except for the Defensive Coverage, Routes, YardLine, Down, PressureOnPlay, and Droptype
* Create dummy variables for the droptype and coverage scheme and the drop the dummy for Spike
* Remove the success\_cat variable for the target
* Split the data into testing, validation, and training sets
* For each of the 3 model types (Decision Tree, Logistic Regression, and SVC) perform the following steps
  + Create list of potential parameters for grid search (i.e. Min\_sample\_split, max\_depth, class\_weight for decision tree)
  + Perform a grid search for each potential parameter combination using the training data and use that best model to predict on the validation set.
  + Using the predictions, evaluate the model, specifically looking at Recall and Balanced Accuracy
  + Look at confusion matrices to see any potential problems or model weaknesses
* Compare the 3 models to determine the best and make predictions using that best model on the testing data
* Visualize the decision tree and the logistic regression coefficients to take a deeper look into the predictions and see whether drop type was used by either model

### Interception Classification

* Drop the following columns: 'OffensiveYardage', 'EPA', 'FumbleByPasser', 'Safety', 'Touchdown', 'FirstDown', 'ThrowAway', 'Spike', 'SackOnPlay', 'PassBreakupOnPlay', 'Completion', 'Turnover', 'FumbleByReceiver', 'KEY', 'EventType', 'GameID\_x', 'EventID\_x', 'Season', 'Week', 'OffensiveTeam', 'DefensiveTeam', 'EventID\_y', 'GameID\_y', 'PlayDesc', 'FumbleByRusher'
* Create dummy variables for SideOfField, DropType and CoverageScheme
* Remove target variable, ‘InterceptionOnPlay’ from dataframe to create X dataframe, and isolate target variable for y dataframe.
* Using the SMOTE package, perform oversampling of the target variable to create balanced classes
* Split into training and testing data
* For each of the three classifiers (Decision Tree, Random Forest, Adaboost) perform the following steps:
  + Create list of parameters for grid search
    - Decision Tree: min\_samples\_split, max\_depth, ccp\_alpha
    - Random Forest: min\_samples\_split, max\_depth, ccp\_alpha
    - Adaboost: learning rate, number of estimators
  + Perform grid search to identify the optimal parameters for each classification method
  + Apply the optimal parameters and fit the classifier with training data, using the testing data, predict the outcome of classes and identify the accuracy. Computer confusion matrix and then find: recall, specificity, precision, balanced accuracy and F1 score.
  + Compare the different classifiers, using a confusion matrix, and identify the top performing classifier
  + On the top performing classifier, apply feature importance to identify the most influential variables.
* Repeat all of the above steps after dropping the following columns, to determine most influential routes and coverages: 'OffTeamScoreBefore', 'DefTeamScoreBefore', 'Quarter', 'TimeLeft', 'Down', 'ToGo', 'StartYard', 'Hash', 'Attempt', 'ThrowDepth', 'RPO', 'PressureOnPlay', 'success', 'Success\_cat', 'SideOfField\_Oppo', 'SideOfField\_Own'

### Defensive success based on coverage schemes

* Copy the processed dataframe into a new variable and drop the following columns 'GameID\_x', 'EventID\_x', 'Season', 'Week', 'OffensiveTeam', 'DefensiveTeam', 'SideOfField', 'OffTeamScoreBefore', 'DefTeamScoreBefore', 'Quarter', 'TimeLeft', 'EventType', 'FirstDown', 'Touchdown', 'Safety', 'Turnover', 'Attempt', 'Completion', 'Spike', 'ThrowAway', 'ThrowDepth', 'PressureOnPlay', 'SackOnPlay', 'PassBreakupOnPlay', 'InterceptionOnPlay', 'FumbleByPasser', 'FumbleByRusher', 'FumbleByReceiver', 'PlayDesc', 'KEY', 'GameID\_y', 'EventID\_y', 'EPA', 'OffensiveYardage', 'Yardline'
* Invert the 'Success' column 1 to 0 and 0 to 1 to reflect the defensive success
* Drop any NA rows
* Create dummy variables of the following columns 'DropType', 'CoverageScheme'
* Copy target variable 'Success' into a new variable and drop 'Success' from the dataframe
* Split dataset into train and test dataset with following parameters stratify=y, random\_state = 0, test\_size= 0.25
* Logistic regression classifier:
  + Use following parameters: cv = 10, random\_state = 0, max\_iter = 10000, solver='lbfgs'
  + Train and predict the model
  + Create a confusion matrix of the test dataset
  + Isolate the predictions, routes, success percentage, and defensive coverages to combine the data into a table for analysis
* SVC classifier:
  + Copy and create a new dataframe with filter "success" == 1 from the dataframe used for logistic regression
  + Drop rows if "CoverageScheme" == 'Other', 'Spike' and 'Combination'
  + Copy target variable 'CoverageScheme' and drop it from the dataframe
  + Factorize 'CoverageScheme'
  + Replace target variable with factorized 'CoverageScheme'
  + Split dataset into train and test dataset with following parameters stratify=y, random\_state = 0, test\_size= 0.25
  + Standardize train and test dataset using StandardScaler
  + Transform train and test dataset using PCA
  + Use GridSearch to Train and predict the model using PCA transformed data
  + Create a classification report of the model
  + Create a confusion matrix of the test dataset
* SVC reduced class classifier:
  + Repeat the SVC classifier steps but add in the following step before factorizing the target variable:
    - Replace and rename CoverageScheme by coverage = {'Cover 0' : 'Cover 1', 'Cover 1' : 'Cover 1', 'Cover 2' : 'Cover 3', 'Cover 3' : 'Cover 3', 'Cover 4' : 'Cover 3', 'Man Cover 2' : 'Cover 1', 'Cover 6' : 'Cover 3','Screen' : 'Screen','Prevent' : 'Prevent', 'Tampa 2' : 'Tampa 2'}

## Environment Requirement

OS: Windows 10/ Mac OS/ Mac M1

Memory (RAM): 1gb

Disk space: 30mb

CPU: 1 Core

GPU: None

Environment: Google Colaboratory/ Jupyter Notebook

## Data Files

Pbp.csv : <https://raw.githubusercontent.com/samhoppen/DSC-540-Group-Project/main/SIS%20PBP.csv>

Game\_info.csv: <https://raw.githubusercontent.com/samhoppen/DSC-540-Group-Project/main/SIS%20Game%20Info.csv>

Player\_positions.csv: <https://raw.githubusercontent.com/samhoppen/DSC-540-Group-Project/main/SIS%20Player%20Positions.csv>