Shariq Mian

Data 622

HW2

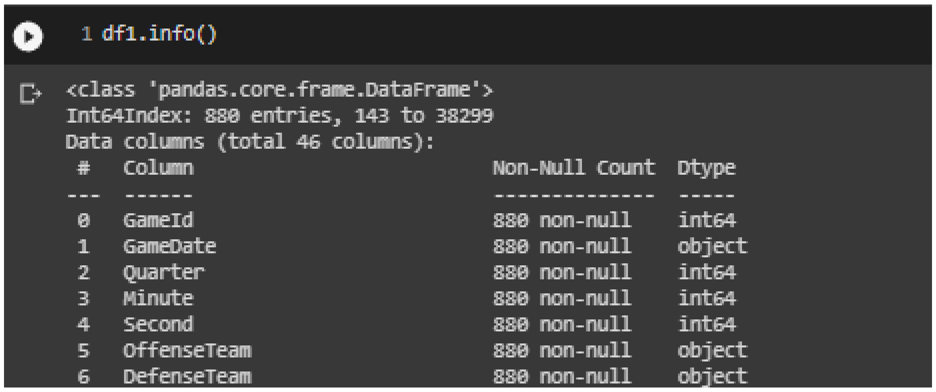
I have been living in NY for 20 years now and as time went on, I have became all NY sports fan. One of my favorite things to do is Watch Football on Sundays and I am a big NY Jets fan. I watch all the games and follow them on social media. In 2022, they started out great but have recently lost 3 out of the last 4 games and are in the brink of not making the playoffs. One thing I have constantly heard from the media/fans is how predictive their play calling is. So, I want to take this opportunity to predict their play calling and whether its some thing we can predict easily and with accuracy.

Data:

I looked at numerous play by play data in order to get the most features possible in order to accurately predict whether a play will be passing vs running play. There was the NGS data set which had many attributes but needed paid subscription for play-by-play data. But the data did include the count of different positions on the field. In the end I used NFL savant data which had all the play by play data for 2022. I specifically chose a single year because personal might change year to year and thus the scheme and play type might change as well. I downloaded and uploaded the data into GitHub in order to read and load via cloud.

Data Exploration:

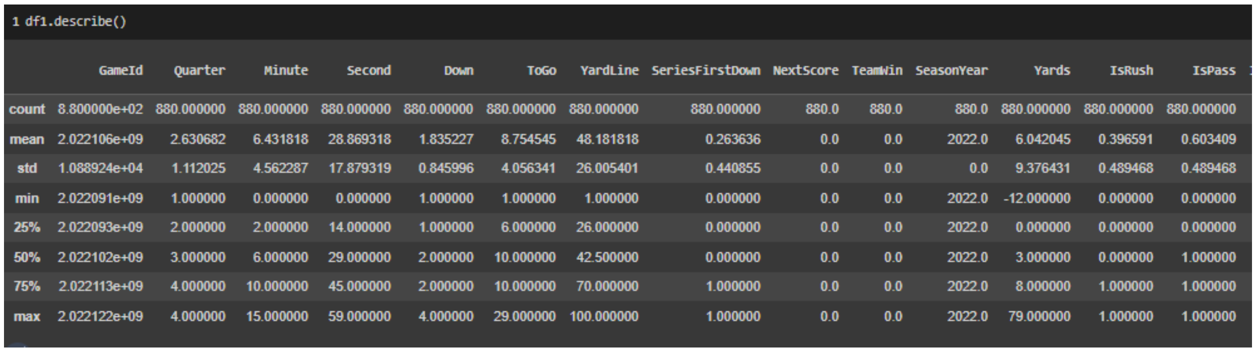
We see the data consist of 46 features and 880 row meaning in 2022, there were 880 plays so far this year for the NY Jets which were either Rush or Pass Plays.



```
1 df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 880 entries, 143 to 38299
Data columns (total 46 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   GameId                          880 non-null    int64
 1   GameDate                        880 non-null    object
 2   Quarter                         880 non-null    int64
 3   Minute                          880 non-null    int64
 4   Second                          880 non-null    int64
 5   OffenseTeam                     880 non-null    object
 6   DefenseTeam                     880 non-null    object
```

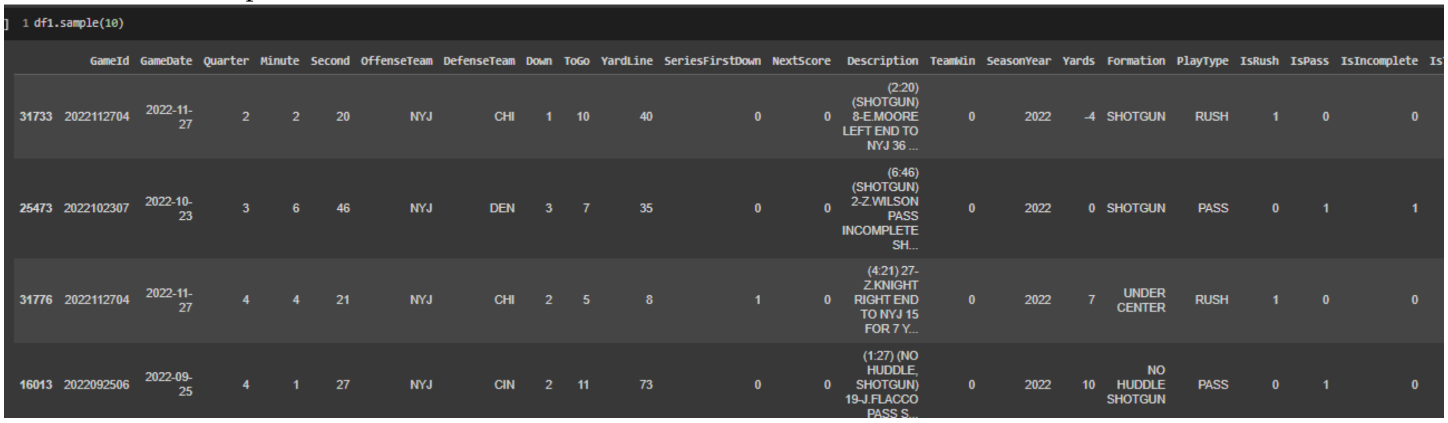
Below is a quick summary of some of the attributes.



```
1 df1.describe()
```

```
             GameId     Quarter      Minute      Second        Down        ToGo    YardLine  SeriesFirstDown  NextScore  TeamWin  SeasonYear       Yards      IsRush      IsPass
count  8.800000e+02  880.000000  880.000000  880.000000  880.000000  880.000000  880.000000       880.000000      880.0    880.0       880.0  880.000000  880.000000  880.000000
mean   2.022106e+09    2.630682    6.431818   28.869318    1.835227    8.754545   48.181818         0.263636        0.0      0.0      2022.0    6.042045    0.396591    0.603409
std    1.088924e+04    1.112025    4.562287   17.879319    0.845996    4.056341   26.005401         0.440855        0.0      0.0         0.0    9.376431    0.489468    0.489468
min    2.022091e+09    1.000000    0.000000    0.000000    1.000000    1.000000    1.000000         0.000000        0.0      0.0      2022.0  -12.000000    0.000000    0.000000
25%    2.022093e+09    2.000000    2.000000   14.000000    1.000000    6.000000   26.000000         0.000000        0.0      0.0      2022.0    0.000000    0.000000    0.000000
50%    2.022102e+09    3.000000    6.000000   29.000000    2.000000   10.000000   42.500000         0.000000        0.0      0.0      2022.0    3.000000    0.000000    1.000000
75%    2.022113e+09    4.000000   10.000000   45.000000    2.000000   10.000000   70.000000         1.000000        0.0      0.0      2022.0    8.000000    1.000000    1.000000
max    2.022122e+09    4.000000   15.000000   59.000000    4.000000   29.000000  100.000000         1.000000        0.0      0.0      2022.0   79.000000    1.000000    1.000000
```

Below is the a sample of the data we have.



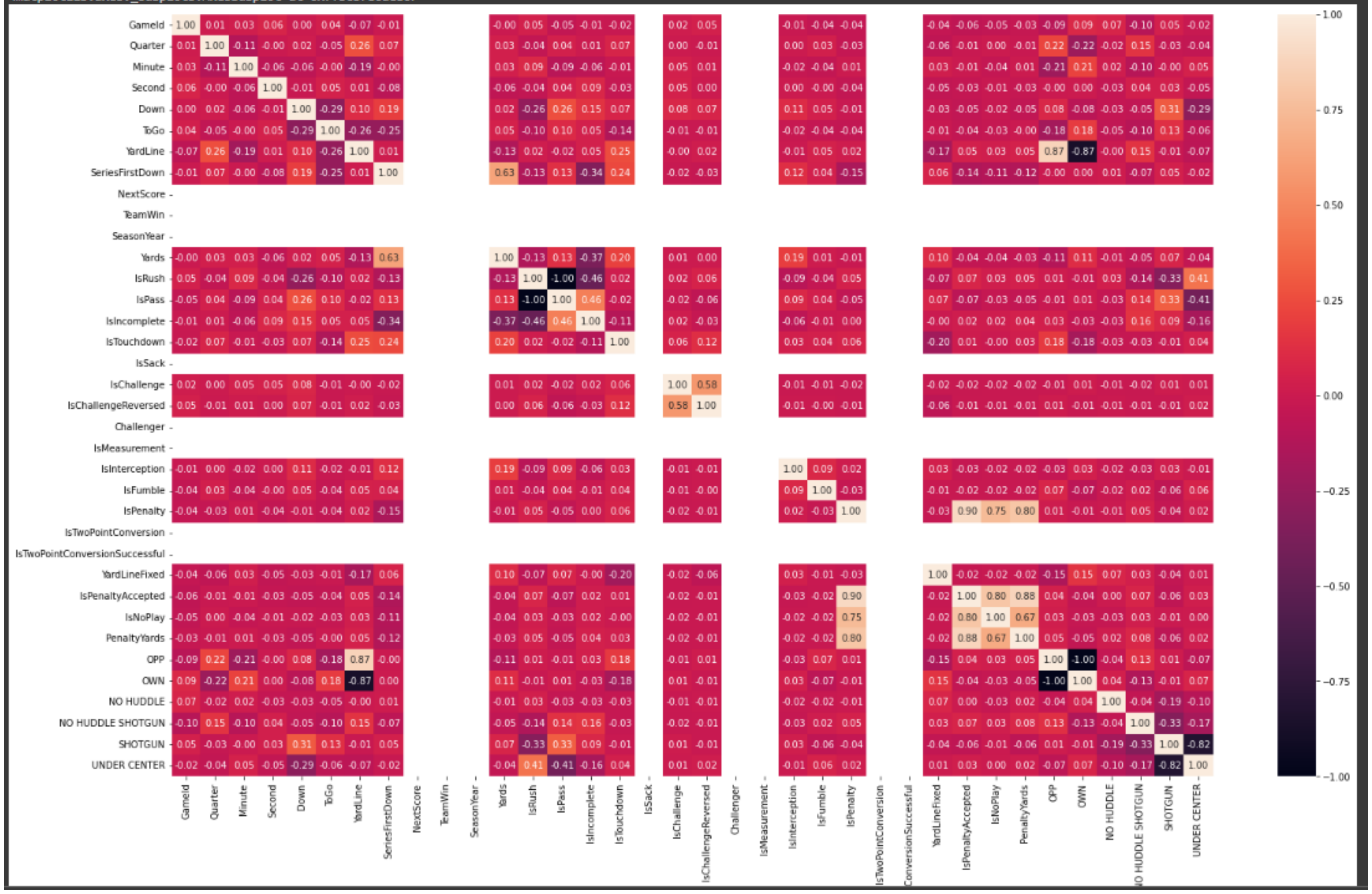
```
df1.sample(10)
```

	GameId	GameDate	Quarter	Minute	Second	OffenseTeam	DefenseTeam	Down	ToGo	YardLine	SeriesFirstDown	NextScore	Description	TeamWin	SeasonYear	Yards	Formation	PlayType	IsRush	IsPass	IsIncomplete
31733	2022112704	2022-11-27	2	2	20	NYJ	CHI	1	10	40	0	0	(2:20) (SHOTGUN) 8-E.MOORE LEFT END TO NYJ 36 ...	0	2022	-4	SHOTGUN	RUSH	1	0	0
25473	2022102307	2022-10-23	3	6	46	NYJ	DEN	3	7	35	0	0	(6:46) (SHOTGUN) 2-Z.WILSON PASS INCOMPLETE SH...	0	2022	0	SHOTGUN	PASS	0	1	1
31776	2022112704	2022-11-27	4	4	21	NYJ	CHI	2	5	8	1	0	(4:21) 27-Z.KNIGHT RIGHT END TO NYJ 15 FOR 7 Y...	0	2022	7	UNDER CENTER	RUSH	1	0	0
16013	2022092506	2022-09-25	4	1	27	NYJ	CIN	2	11	73	0	0	(1:27) (NO HUDDLE, SHOTGUN) 19-J.FLACCO PASS S...	0	2022	10	NO HUDDLE SHOTGUN	PASS	0	1	0

Below is the correlation Matrix. This is where we removed most of the features due to multicollinearity.

```
df1.corr()
plt.figure(figsize=(25 , 15))
sns.heatmap(df1.corr(), annot=True, fmt=".2f")
```

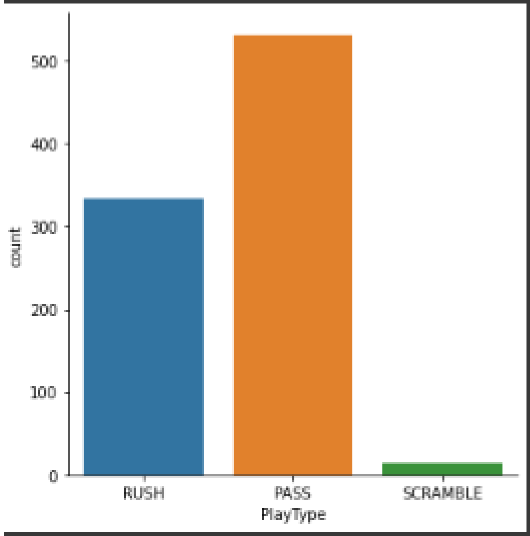
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c3780ae80>
```



Lets look into some details on some of the attributes.

We see that in 2022, the NYJ are predominantly a Pass first offense with almost 200 more pass plays than running plays. This is important because it means the distribution between both play types is not a 50/50 split. By default, there should be slightly more passing plays than running plays for every split. Scrambles are passes which becomes a rush because the defense was close to getting to the quarterback and the Quarterback scrambles for some yardage.

```python
sns.catplot(x='PlayType', kind='count', data=df1, orient='h')
plt.show()
```

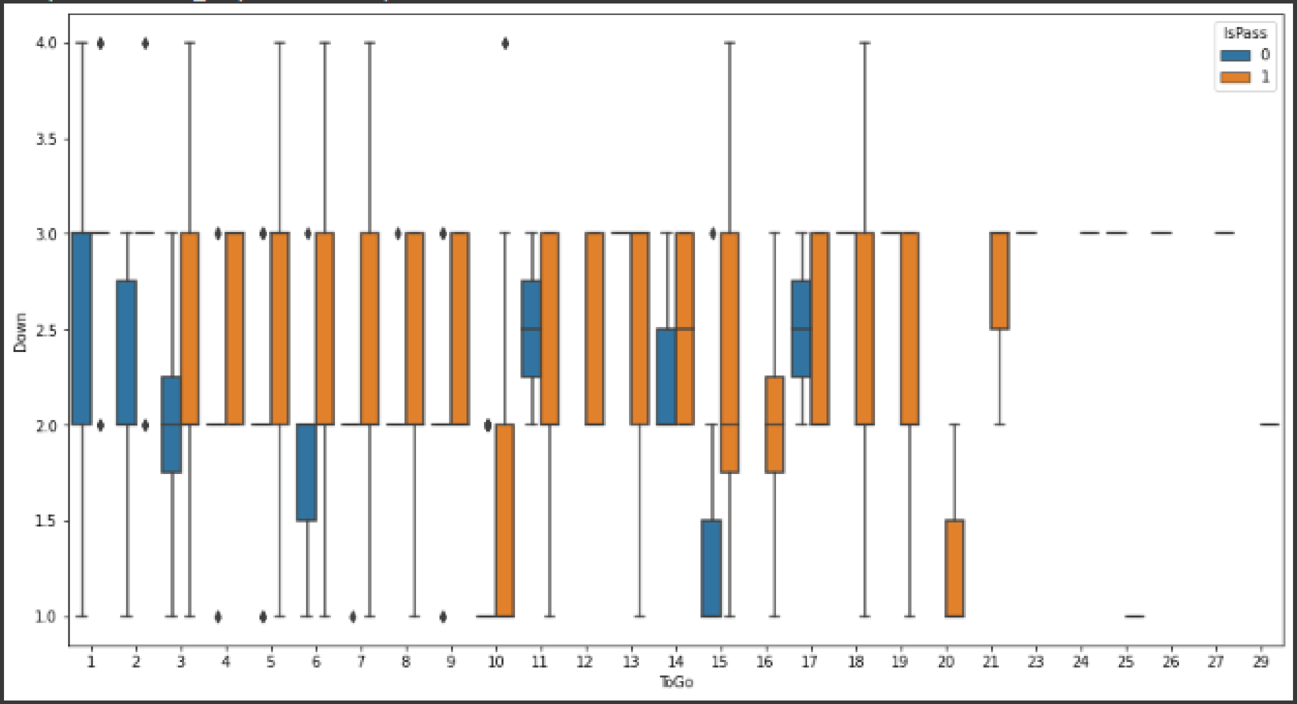


A down is a period where a team can attempt a play. In the NFL, an offense gets four play attempts (called "downs") to gain a specified number of yards (usually starting with 10 yards); if it doesn't, it has to give the ball to the opponent.

Next in the boxplot we see 3rd downs are predominantly pass plays when the yardage is above 3 yards. but, given the initial data distribution, this is probably meaningless.

```python
plt.figure(figsize=(15, 8))
sns.boxplot(x='ToGo', y="Down",hue='IsPass', data=df1)
```

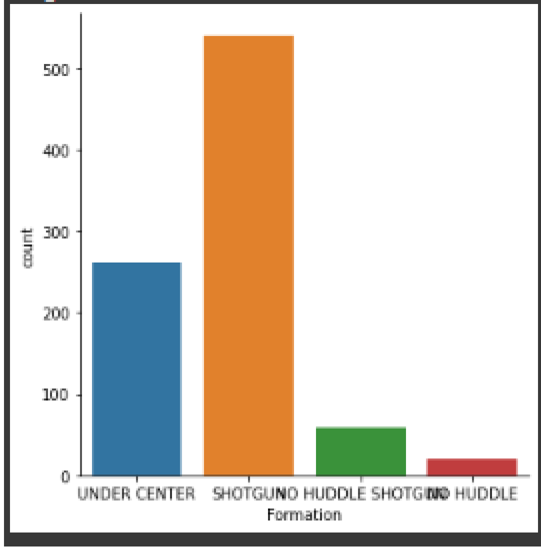
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c37e99670>
```



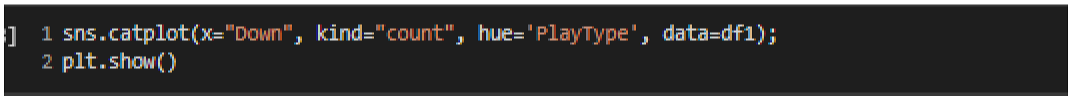
We further see that play type based on formations. We see that almost 520 plays were predominantly from Shotgun formation, which makes sense because Shotgun formation is normally a passing formation as it gives the quarterback more time to throw.

```
plt.figure(figsize=(10, 8))
sns.catplot(x='Formation', kind='count', data=df1, orient='h')
plt.show()
```

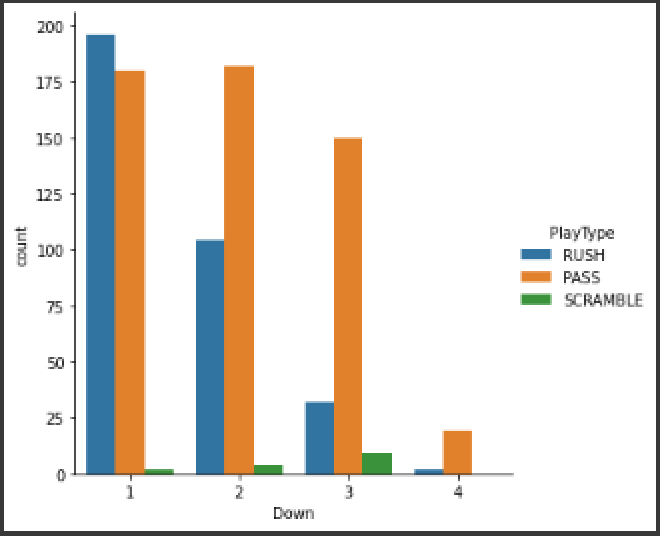
```
<Figure size 720x576 with 0 Axes>
```



Next we have a breakdown of Downs against the pass Play type, We see that on 1st down there are almost equal amount of changes of a running play vs passing play, while on 2nd down it tends to be more passing plays while on third down its almost always a passing play. This looks like all the information we need to predict a play however there is more to predicting play calling than this. Time and the position of the ball on the field matter as well. When you have low yardage its better to run vs pass. Is your opponent is on their 3 yard line, how do you decide whether the play will be run vs pass. Rushes dominate at the Offense's endzone, whereas they tend to Pass more frequently (much more frequently) when facing the goal line. This shows the relation between the risk the QB of the Offense has in his own endzone, versus when they're knocking on the door of the Defense. Formations likely have a lot to do with the endzone success as well, so I'm interested to see how they show their importance in the prediction model.

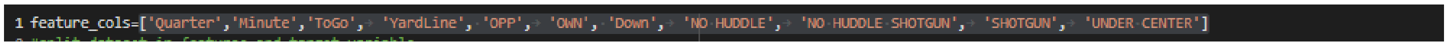


```
sns.catplot(x="Down", kind="count", hue='PlayType', data=df1);
plt.show()
```



Decision Tree Model 1

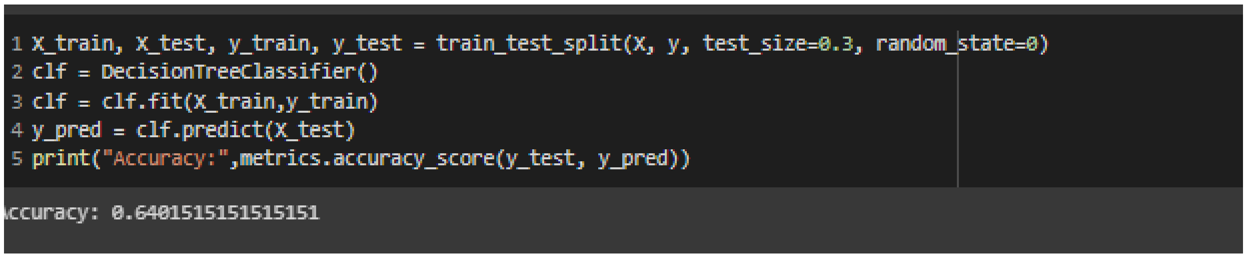
Based on my exploration, and the relevance I saw in those visualizations, among many, many others, I chose to focus on the features below for the first decision tree model. A lot of the features were the results of a play and not information we would know before the play thus the reason for low features,



```
feature_cols=['Quarter','Minute','ToGo', 'YardLine', 'OPP', 'OWN', 'Down', 'NO HUDDLE', 'NO HUDDLE SHOTGUN', 'SHOTGUN', 'UNDER CENTER']
```

Some interesting things I did not include in my models were that Seconds had almost no predictive power, but I felt it necessary to keep checking it throughout my models just to make sure I wasn’t missing secret information.

For the 1st model got an accuracy of 64.02% which was not bad for an initial model.

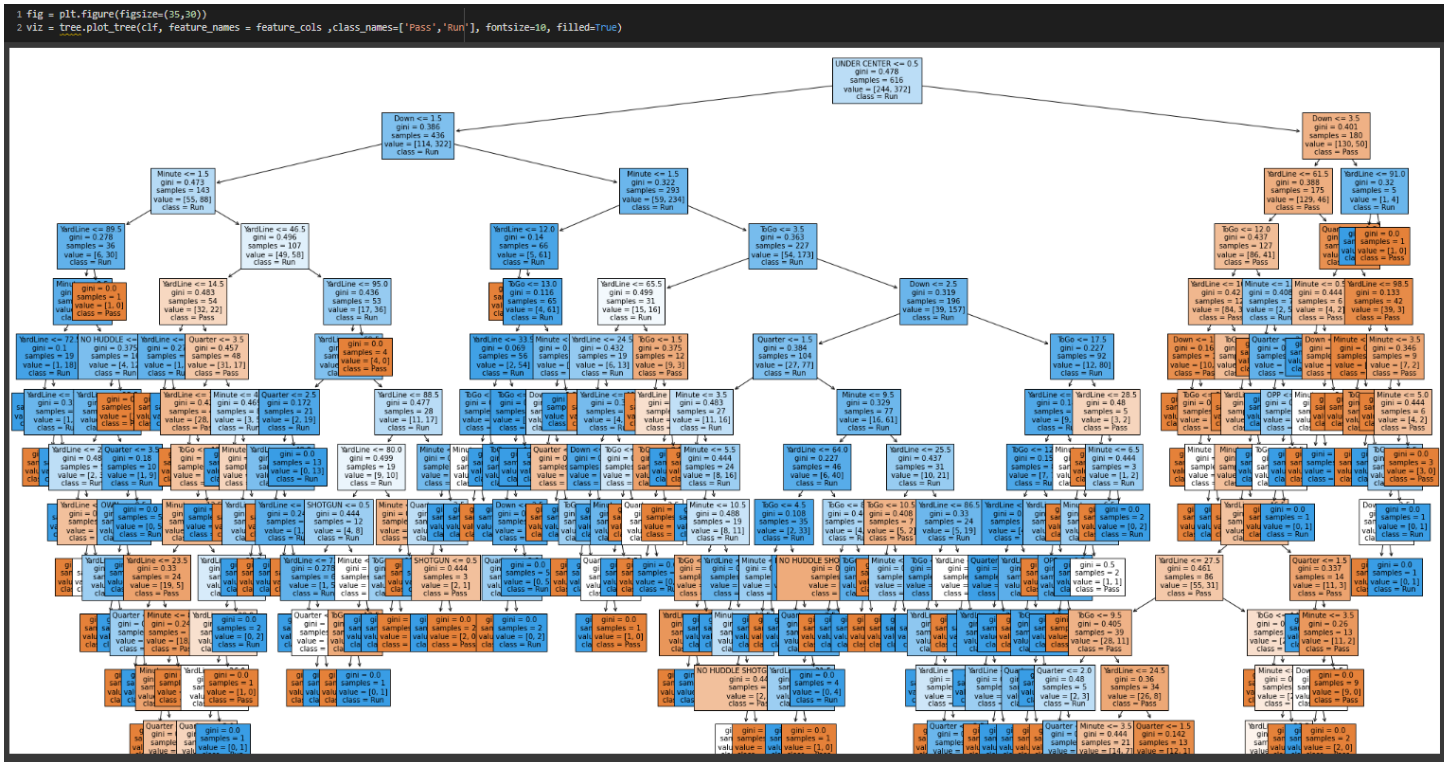


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
ccuracy: 0.6401515151515151
```

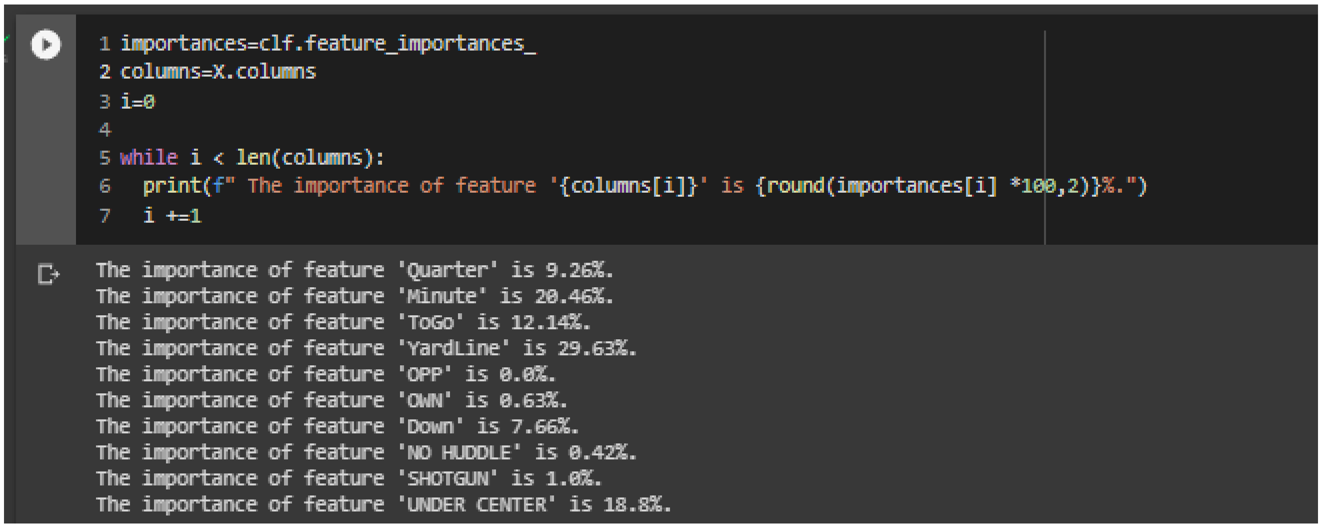
And our initial tree looks very intimidating.

Right of the bat, Formation has high importance here in indicating whether or not the ball would be rushed. The way the team lined up, it seems, gave great predictive power for when the team would Rush, and was used to help the model discern how to tell the difference. we see the first split is Formation (under Center) with the second split being Down < 1.5 . Next split Downs, Minute, yard line, and ToGo (Yards the Offense had left to get a first down) had the highest positive impact on our prediction capabilities.



```
fig = plt.figure(figsize=(35,30))
viz = tree.plot_tree(clf, feature_names = feature_cols ,class_names=['Pass','Run'], fontsize=10, filled=True)
```

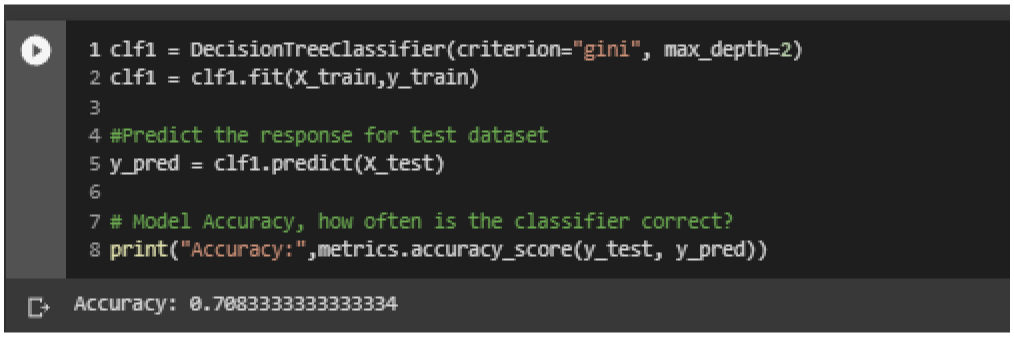
Below is the importance of each feature that we used in the decision tree. We see that Yard line has the biggest importance. This makes sense because if you are on your own side of the field and deep with in your territory you tent to run the ball instead of the passing. And when you are deep in your opponent territory within in 5 yard line you always tend to rush the ball.



```
importances=clf.feature_importances_
columns=X.columns
i=0

while i < len(columns):
  print(f" The importance of feature '{columns[i]}' is {round(importances[i] *100,2)}%.")
  i +=1
```

```
The importance of feature 'Quarter' is 9.26%.
The importance of feature 'Minute' is 20.46%.
The importance of feature 'ToGo' is 12.14%.
The importance of feature 'YardLine' is 29.63%.
The importance of feature 'OPP' is 0.0%.
The importance of feature 'OWN' is 0.63%.
The importance of feature 'Down' is 7.66%.
The importance of feature 'NO HUDDLE' is 0.42%.
The importance of feature 'SHOTGUN' is 1.0%.
The importance of feature 'UNDER CENTER' is 18.8%.
```

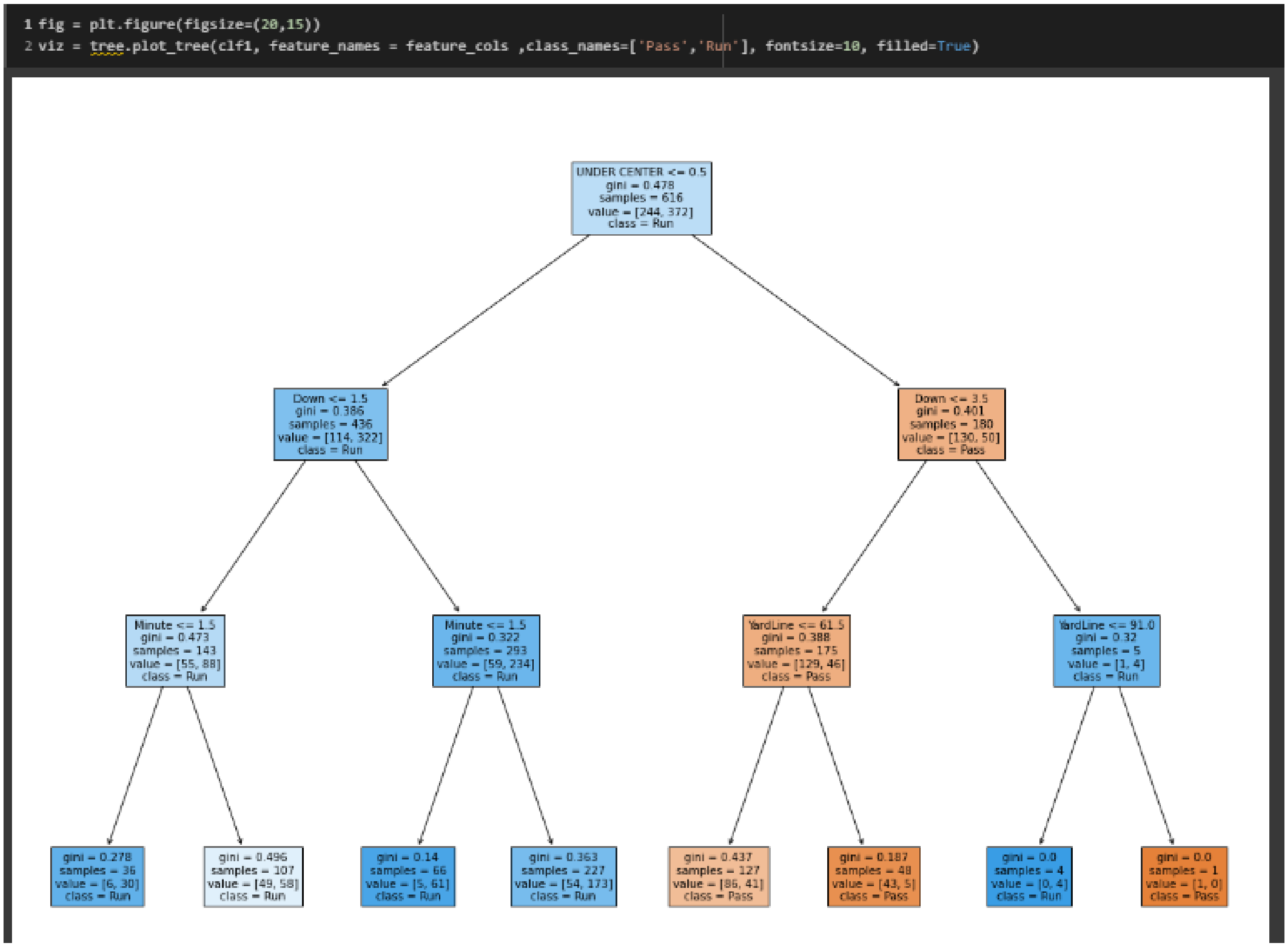
Shallow Decision Tree is the best baseline classification model a data scientist could ask for, and they were right. By starting with a shallow tree with a max depth of 3 (as seen below), we immediately had a way of beating the majority baseline, and an easy way to check my model for leakage. Here we get an accuracy of 70.83% for depth of 3.



```
clf1 = DecisionTreeClassifier(criterion="gini", max_depth=2)
clf1 = clf1.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf1.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7083333333333334
```

Here we see the actual tree with depth of 3 and the nodes result. On one side we have a branch which is predominantly run situations when Down < 1 and minute <1.5 then those are run situation while on the right we have down <3.5 and Yard line <61.5 then pass but Yard line <91 then predicting a Run.

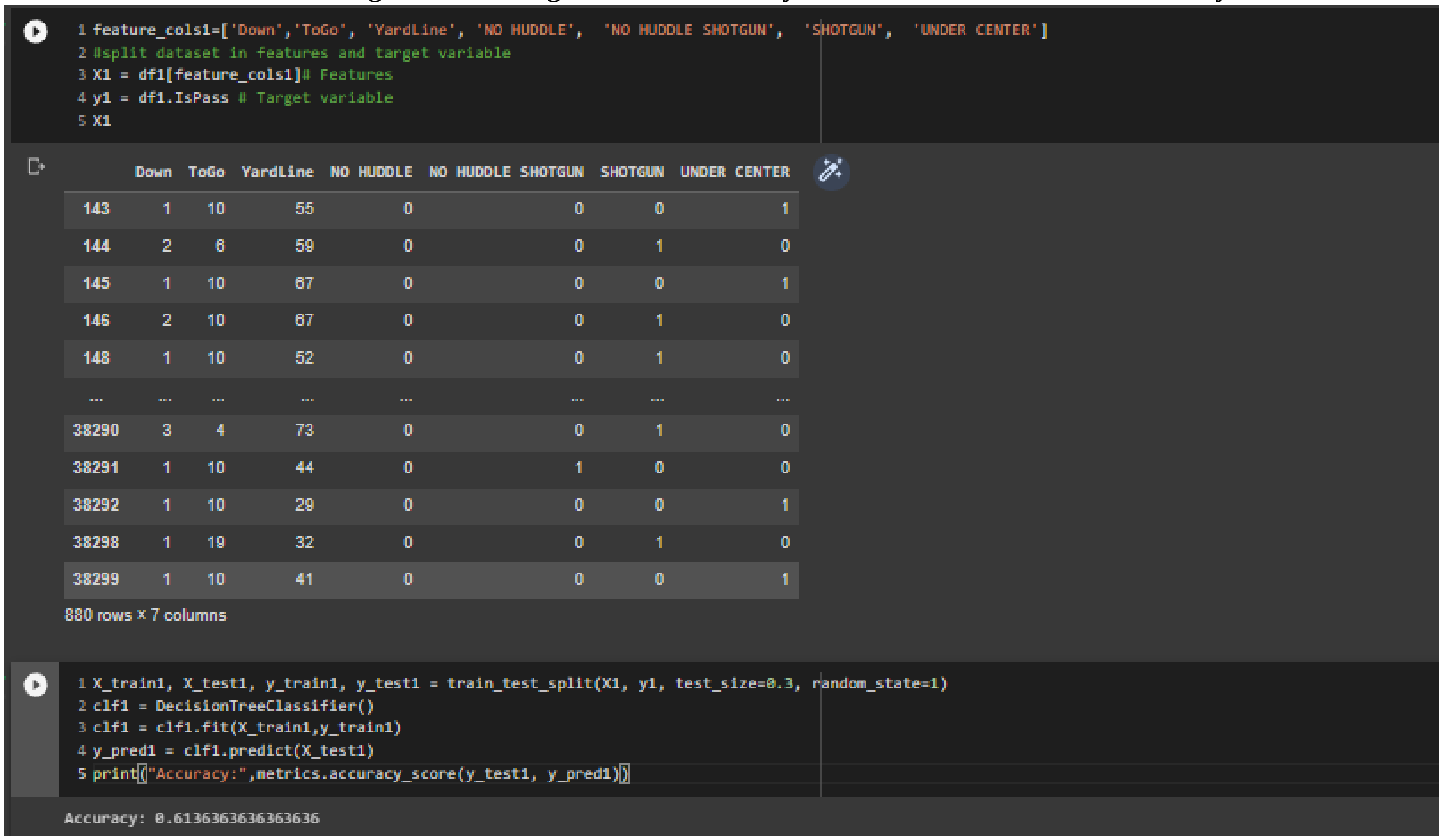


```
fig = plt.figure(figsize=(20,15))
viz = tree.plot_tree(clf1, feature_names = feature_cols ,class_names=['Pass','Run'], fontsize=10, filled=True)
```

UNDER CENTER <= 0.5 gini = 0.478 samples = 616 value = [244, 372] class = Run
Down <= 1.5 gini = 0.386 samples = 436 value = [114, 322] class = Run
Down <= 3.5 gini = 0.401 samples = 180 value = [130, 50] class = Pass
Minute <= 1.5 gini = 0.473 samples = 143 value = [55, 88] class = Run
Minute <= 1.5 gini = 0.322 samples = 293 value = [59, 234] class = Run
YardLine <= 61.5 gini = 0.388 samples = 175 value = [129, 46] class = Pass
YardLine <= 91.0 gini = 0.32 samples = 5 value = [1, 4] class = Run
gini = 0.278 samples = 36 value = [6, 30] class = Run
gini = 0.496 samples = 107 value = [49, 58] class = Run
gini = 0.14 samples = 66 value = [5, 61] class = Run
gini = 0.363 samples = 227 value = [54, 173] class = Run
gini = 0.437 samples = 127 value = [86, 41] class = Pass
gini = 0.187 samples = 48 value = [43, 5] class = Pass
gini = 0.0 samples = 4 value = [0, 4] class = Run
gini = 0.0 samples = 1 value = [1, 0] class = Pass

Decision Tree Model 2

We recreate another model but with different variables. Here we removeQuarter, Minutesm Opp, OWN, No Huddle and No Huddle Shotgun. Here we get a lower accuacy than the first model. An accuarcy of 61.36%.



```
feature_cols1=['Down','ToGo', 'YardLine', 'NO HUDDLE',  'NO HUDDLE SHOTGUN',  'SHOTGUN',  'UNDER CENTER']
#split dataset in features and target variable
X1 = df1[feature_cols1]# Features
y1 = df1.IsPass # Target variable
X1
```

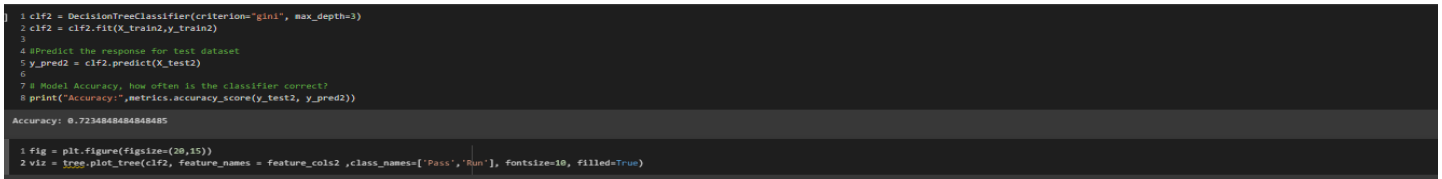
	Down	ToGo	YardLine	NO HUDDLE	NO HUDDLE SHOTGUN	SHOTGUN	UNDER CENTER
143	1	10	55	0	0	0	1
144	2	6	59	0	0	1	0
145	1	10	67	0	0	0	1
146	2	10	67	0	0	1	0
148	1	10	52	0	0	1	0
...	...	...	...	...	...	...	...
38290	3	4	73	0	0	1	0
38291	1	10	44	0	1	0	0
38292	1	10	29	0	0	0	1
38298	1	19	32	0	0	1	0
38299	1	10	41	0	0	0	1

880 rows × 7 columns

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.3, random_state=1)
clf1 = DecisionTreeClassifier()
clf1 = clf1.fit(X_train1,y_train1)
y_pred1 = clf1.predict(X_test1)
print("Accuracy:",metrics.accuracy_score(y_test1, y_pred1))
```

```
Accuracy: 0.6136363636363636
```

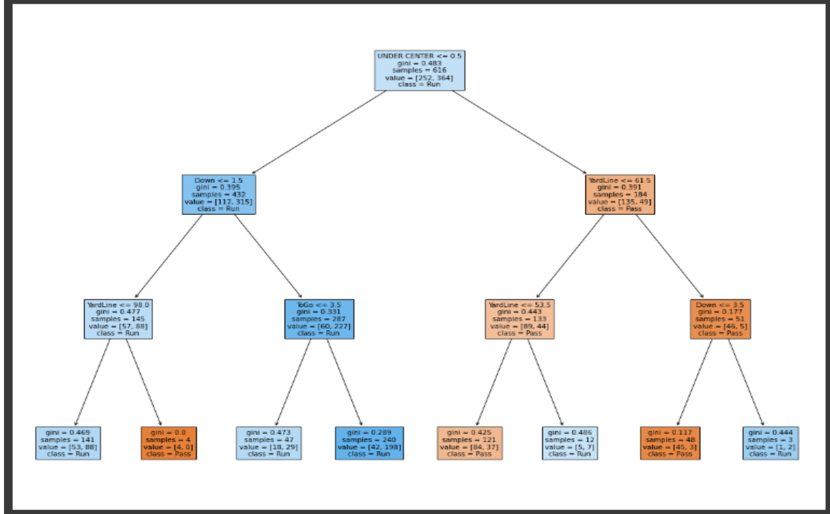
When we apply the depth of 3 we see that this model has an accuracy of 72.34%. With If not under center and down if its 1st down and its less than 98 yard line then Run or if to go is less than 3.5 yards then run play is predicted while if you are under center and the yard line is less than 61.5 and down is less than 3.5 then you predict pass play.



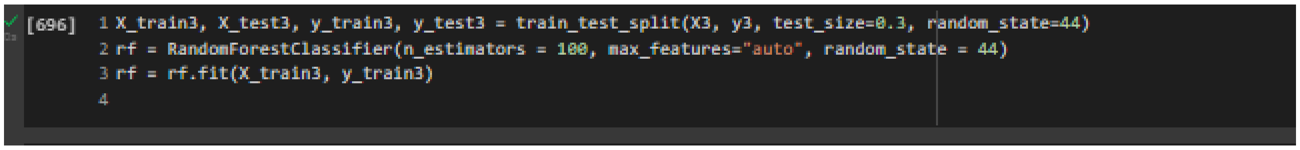
```
clf2 = DecisionTreeClassifier(criterion="gini", max_depth=3)
clf2 = clf2.fit(X_train2,y_train2)

#Predict the response for test dataset
y_pred2 = clf2.predict(X_test2)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test2, y_pred2))
```

```
Accuracy: 0.7234848484848485
```

```
fig = plt.figure(figsize=(20,15))
viz = tree.plot_tree(clf2, feature_names = feature_cols2 ,class_names=['Pass','Run'], fontsize=10, filled=True)
```

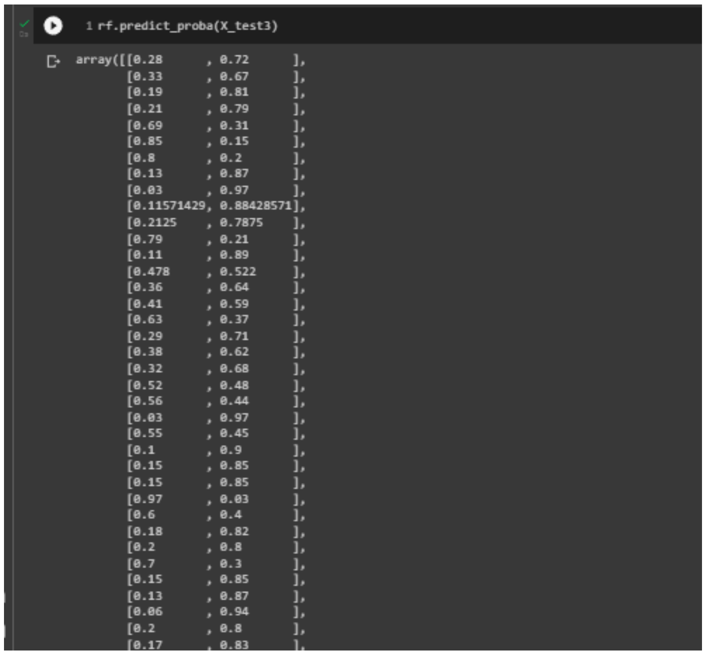


Model 3 Random Forest:
We now use Random Forest to predict our pass vs run plays. We are using 100 N Estimators.



```
X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3, test_size=0.3, random_state=44)
rf = RandomForestClassifier(n_estimators = 100, max_features="auto", random_state = 44)
rf = rf.fit(X_train3, y_train3)
```

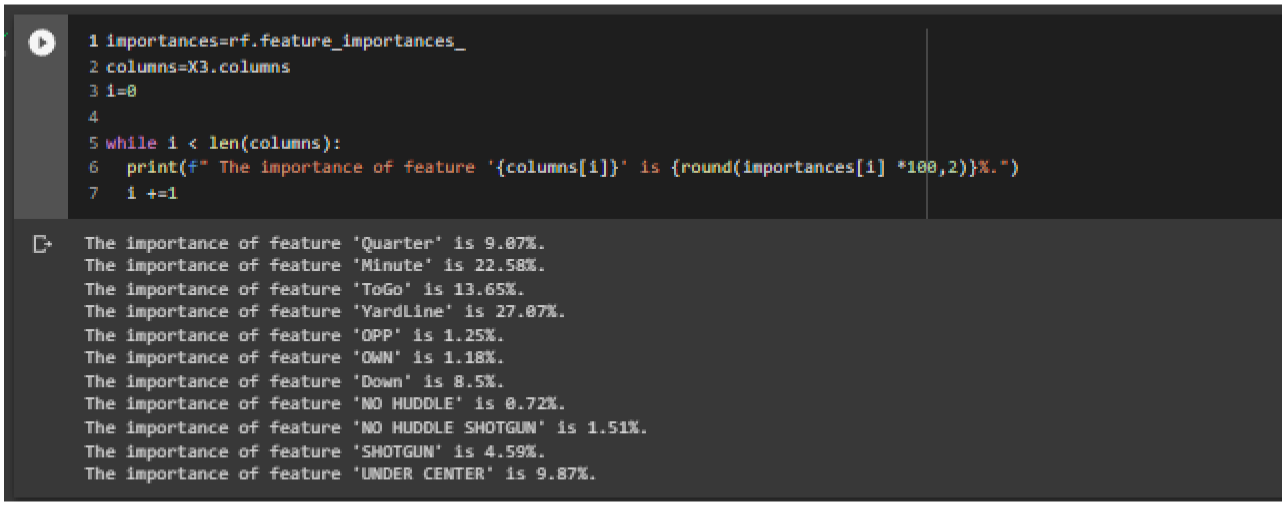
Below are some of the predictions that we have for pass vs run plays,



```
rf.predict_proba(X_test3)
```

```
array([[0.28      , 0.72      ],
       [0.33      , 0.67      ],
       [0.19      , 0.81      ],
       [0.21      , 0.79      ],
       [0.69      , 0.31      ],
       [0.85      , 0.15      ],
       [0.8       , 0.2       ],
       [0.13      , 0.87      ],
       [0.03      , 0.97      ],
       [0.11571429, 0.88428571],
       [0.2125    , 0.7875    ],
       [0.79      , 0.21      ],
       [0.11      , 0.89      ],
       [0.478     , 0.522     ],
       [0.36      , 0.64      ],
       [0.41      , 0.59      ],
       [0.63      , 0.37      ],
       [0.29      , 0.71      ],
       [0.38      , 0.62      ],
       [0.32      , 0.68      ],
       [0.52      , 0.48      ],
       [0.56      , 0.44      ],
       [0.03      , 0.97      ],
       [0.55      , 0.45      ],
       [0.1       , 0.9       ],
       [0.15      , 0.85      ],
       [0.15      , 0.85      ],
       [0.97      , 0.03      ],
       [0.6       , 0.4       ],
       [0.18      , 0.82      ],
       [0.2       , 0.8       ],
       [0.7       , 0.3       ],
       [0.15      , 0.85      ],
       [0.13      , 0.87      ],
       [0.06      , 0.94      ],
       [0.2       , 0.8       ],
       [0.17      , 0.83      ],
```

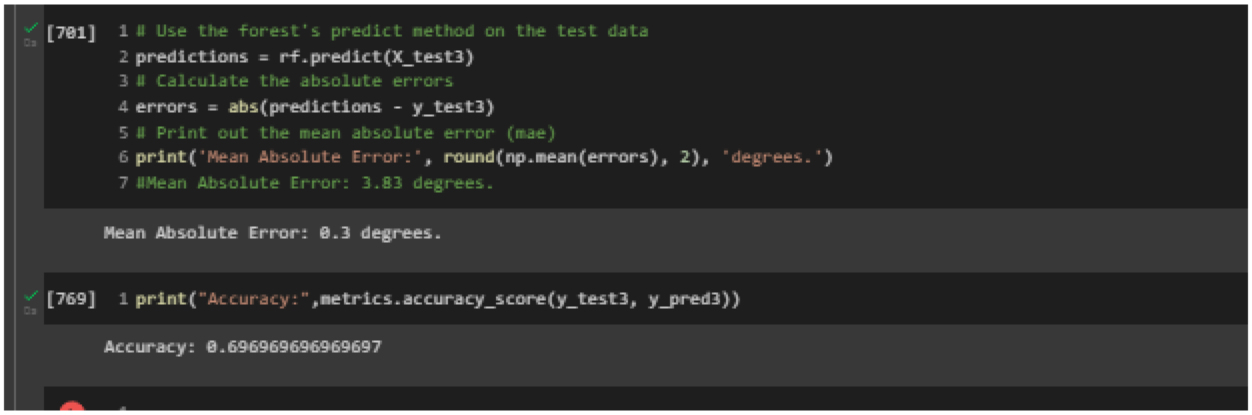
Here is the list of importance of the features used in the Random Forest



```
importances=rf.feature_importances_
columns=X3.columns
i=0

while i < len(columns):
  print(f" The importance of feature '{columns[i]}' is {round(importances[i] *100,2)}%.")
  i +=1
```

```
The importance of feature 'Quarter' is 9.07%.
The importance of feature 'Minute' is 22.58%.
The importance of feature 'ToGo' is 13.65%.
The importance of feature 'YardLine' is 27.07%.
The importance of feature 'OPP' is 1.25%.
The importance of feature 'OWN' is 1.18%.
The importance of feature 'Down' is 8.5%.
The importance of feature 'NO HUDDLE' is 0.72%.
The importance of feature 'NO HUDDLE SHOTGUN' is 1.51%.
The importance of feature 'SHOTGUN' is 4.59%.
The importance of feature 'UNDER CENTER' is 9.87%.
```

For Random Forest we get an MAE of .3 Degrees with an accuracy of 69.70% which is 5.68 % greater than our initial decision tree model 1 with no pruning.



```
[701]
# Use the forest's predict method on the test data
predictions = rf.predict(X_test3)
# Calculate the absolute errors
errors = abs(predictions - y_test3)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
#Mean Absolute Error: 3.83 degrees.
```

```
Mean Absolute Error: 0.3 degrees.
```

```
[769]
print("Accuracy:",metrics.accuracy_score(y_test3, y_pred3))
```

```
Accuracy: 0.696969696969697
```

Conclusion:

In reading the article 'the bad & ugly' aspects of decision trees and thinking about the dataset I chose; I find the decision trees most helpful in this case. First, coaching in sports where you are on the sideline is all about the decisions and making those in split seconds. The results verified my assumption that, Formation, Yards to Go and Yard line are the biggest predictor of where a play is Run or Pass. While these were important other factors as Quarter and Minutes were not as impactful as what I had predicted. One of the bad that I felt occurred in this project was the complexity of the tree where it was hard to understand a branch from the top until the end node in the full model. While the ugly problem was the usability. I find it hard to use this decision trees as one I was unable to get binary results in some of the feature. I think it would have been better to have 1 vs 2nd vs 3rd vs 4th down and the decision based on that but instead it was treated as a continuous variable even though there was ordinality to the variable and thus <1.5 down is not really a great split for this categorical data. Lastly, I want to comment on the fact that the accuracy was low. I believe access to the number of players per position on the field can greatly increase the accuracy and thus is some thing I will be looking to integrate into this prediction in the future.

Python Notebook with Code:

https://colab.research.google.com/drive/1Js3V9Pe_lnFC8Hgf1wgpRKklS-kdBWny?usp=sharing