DSC630-T302: Brandon Mather - Second/Final Model Attempt

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
```

In [3]:
```
offense2022 = pd.read_csv('2022_offensegames.csv')
offense2021 = pd.read_csv('2021_offensegames.csv')
offense2020 = pd.read_csv('2020_offensegames.csv')
offense2019 = pd.read_csv('2019_offensegames.csv')
offense2018 = pd.read_csv('2018_offensegames.csv')
defense2022 = pd.read_csv('2022_defensegames.csv')
defense2021 = pd.read_csv('2021_defensegames.csv')
defense2020 = pd.read_csv('2020_defensegames.csv')
defense2019 = pd.read_csv('2019_defensegames.csv')
defense2018 = pd.read_csv('2018_defensegames.csv')
```

In [4]:
```
offensive_stats = pd.concat([offense2022, offense2021, offense2020, offense2019, offen
defensive_stats = pd.concat([defense2022, defense2021, defense2020, defense2019, defen
```

In [5]:
```
#Dropping stats not needed for model
to_drop_offense = ['Week','Day', 'Date', 'Unnamed: 3', 'OT', 'Unnamed: 6', 'Opp', 'Cmp
                   'NY/A','Cmp%', 'Rate', 'Y/A', 'FGM', 'FGA', 'XPM', 'XPA', 'Pnt', 'Y
to_drop_defense = ['Week','Day', 'Date', 'Unnamed: 3', 'Unnamed: 4', 'OT', 'Unnamed: 6
                   'NY/A','Cmp%', 'Rate', 'Y/A', 'FGM', 'FGA', 'XPM', 'XPA', 'Pnt', 'Y
```

In [6]:
```
offensive_stats.drop(to_drop_offense, inplace=True, axis=1)
defensive_stats.drop(to_drop_defense, inplace=True, axis=1)
```

In [7]:
```
to_drop_offense2 = ['Att.1', 'Y/A.1', 'Yds.1']
to_drop_defense2 = ['Att.1', 'Y/A.1', 'Yds.1']
```

In [8]:
```
offensive_stats.drop(to_drop_offense2, inplace=True, axis=1)
defensive_stats.drop(to_drop_defense2, inplace=True, axis=1)
```

In [9]:
```
#Renaming columns so they are easy to understand
offensive_stats.rename(columns = {'Unnamed: 4':'Result', 'Tm':'Points Scored', 'Opp.1'
                                  'PassYds':'Passing Yards', 'PassTD':'Passing Touchdow
                                  'RunYds': 'Run Yards', 'RunTD': 'Run Touchdowns'}, inp
defensive_stats.rename(columns = {'Unnamed: 4':'Result', 'PassYds':'Passing Yards Agai
                                'Int':'Interceptions', 'RunYds': 'Run Yards Against', 'R
```

In [10]:
```
overall_stats = pd.concat([offensive_stats, defensive_stats], axis=1)
```

In [11]:

```
overall_stats
```

Out[11]:

	Result	Points Scored	Points Scored Against	Passing Yards	Passing Touchdowns	Interceptions Thrown	Run Yards	Run Touchdowns	Passing Yards Against	P Touch A
0	L	7	23	227	0	1	111	1	269	
1	W	27	10	211	2	0	203	1	48	
2	W	14	12	248	2	1	67	0	251	
3	W	27	24	244	2	1	199	1	104	
4	L	22	27	207	2	0	94	0	213	
...	...	...	...	...	...	...	...	...	...	
11	L	17	20	227	1	0	98	1	133	
12	W	34	20	162	2	0	138	1	237	
13	L	17	24	235	0	1	88	1	232	
14	W	44	38	413	2	0	127	3	323	
15	L	0	31	129	0	1	46	0	272	

82 rows × 13 columns

Logistic Regression Model

In [91]:

```
Y = overall_stats.Result
X = overall_stats.drop('Result', axis = 1)
```

In [92]:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state
```

In [93]:

```
pipe = make_pipeline(StandardScaler(), LogisticRegression())
```

In [94]:

```
pipe.fit(x_train, y_train)
```

Out[94]:

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
```

In [96]:

```
y_pred = pipe.predict(x_test)
```

In [95]:

```
pipe.score(x_test, y_test)
```

Out[95]:

```
0.8823529411764706
```

In [97]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           L       1.00      0.71      0.83         7
           W       0.83      1.00      0.91        10

    accuracy                           0.88        17
   macro avg       0.92      0.86      0.87        17
weighted avg       0.90      0.88      0.88        17
```

Random Forest

In [98]:

```
rf = RandomForestClassifier()
```

In [99]:

```
rf.fit(x_train, y_train)
```

Out[99]:

```
RandomForestClassifier()
```

In [100…

```
y_pred_rf = rf.predict(x_test)
```

In [101…

```
rf.score(x_test, y_test)
```

Out[101]:

```
0.9411764705882353
```

In [102…

```
print(classification_report(y_test,y_pred_rf))
```

```
              precision    recall  f1-score   support

           L       1.00      0.86      0.92         7
           W       0.91      1.00      0.95        10

    accuracy                           0.94        17
   macro avg       0.95      0.93      0.94        17
weighted avg       0.95      0.94      0.94        17
```

Feature Importance for Logistic Regression Model

In [104…

```
feature_names = x_train.columns
```

In [105…

```
coefs = pipe.named_steps["logisticregression"].coef_.flatten()
```

In [106…

```
zipped = zip(feature_names, coefs)
```

In [107…

```
df = pd.DataFrame(zipped, columns=["feature", "value"])
```

In [108…

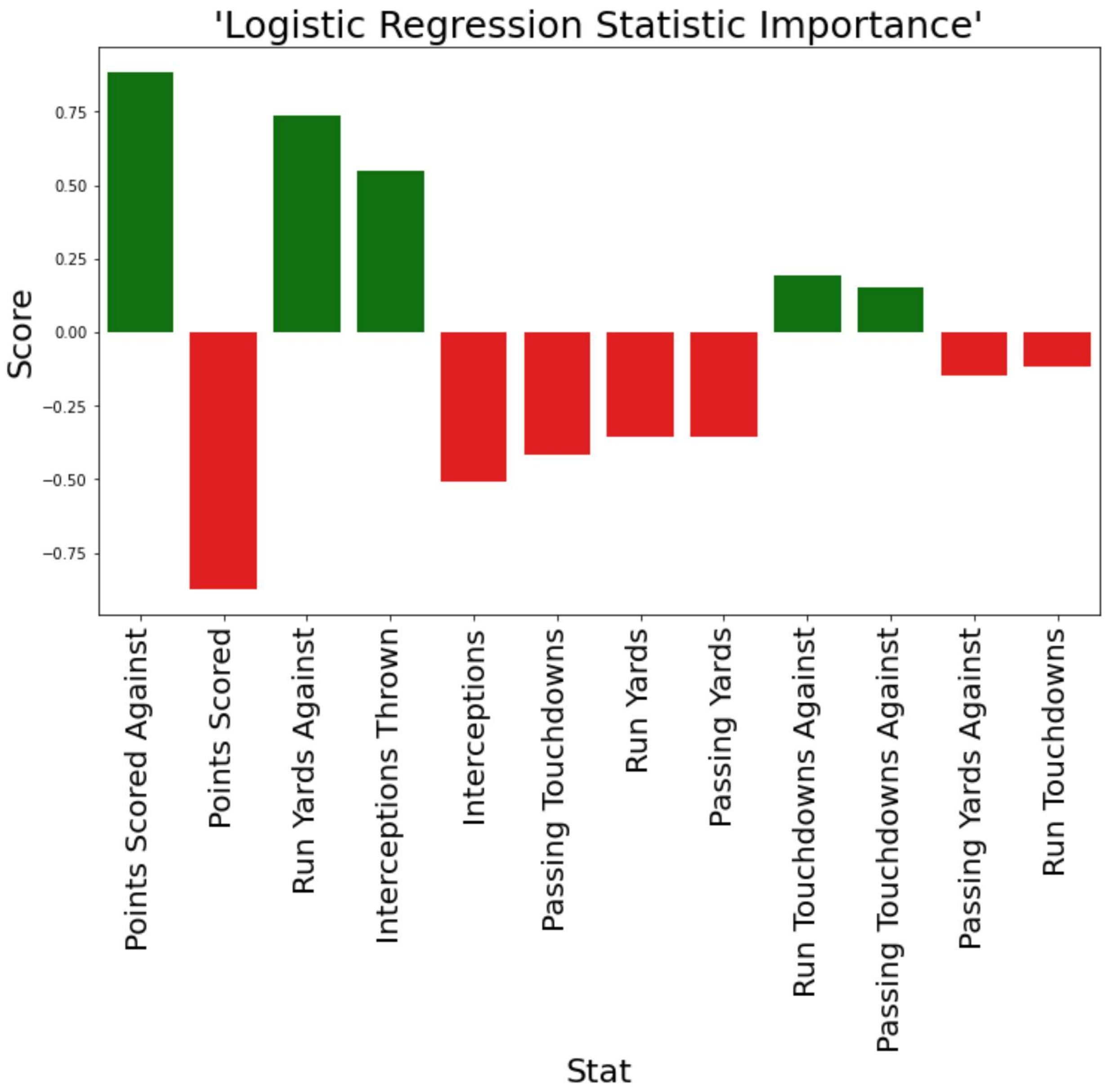
```
df["abs_value"] = df["value"].apply(lambda x: abs(x))
df["colors"] = df["value"].apply(lambda x: "green" if x > 0 else "red")
df = df.sort_values("abs_value", ascending=False)
```

In [117…

```
fig, ax = plt.subplots(1, 1, figsize=(12, 7))
sns.barplot(x="feature",
            y="value",
            data=df,
           palette=df["colors"])
ax.set_xticklabels(ax.get_xticklabels(), rotation=90, fontsize=20)
ax.set_title("'Logistic Regression Statistic Importance'", fontsize=25)
ax.set_ylabel("Score", fontsize=22)
ax.set_xlabel("Stat", fontsize=22)
```

Out[117]:

```
Text(0.5, 0, 'Stat')
```



Feature Importance for Random Forest Model

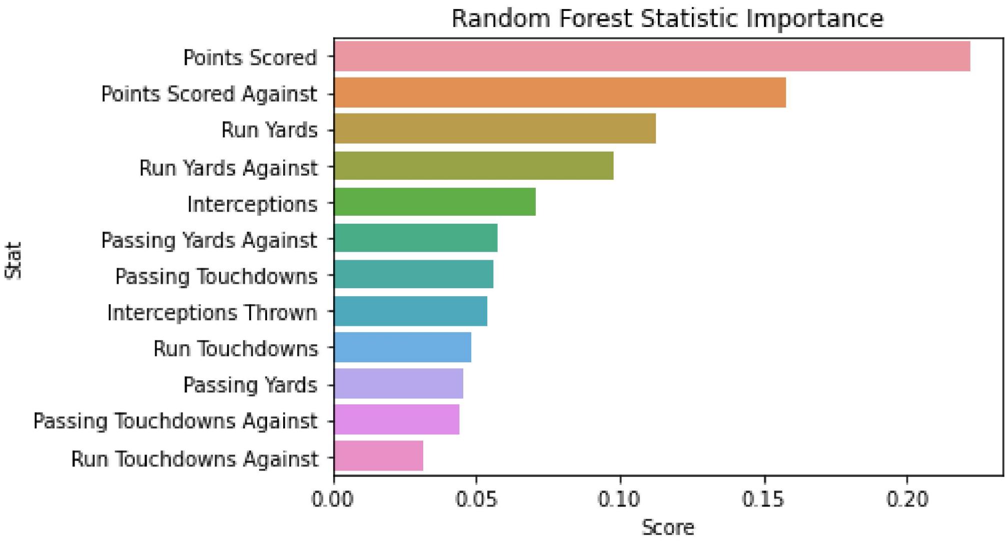
In [114…

```
classifiers = pd.Series(rf.feature_importances_, index=x_train.columns)
classifiers.sort_values(ascending=False, inplace=True)
```

In [116…

```
sns.barplot(x = classifiers, y = classifiers.index)

plt.xlabel('Score')
plt.ylabel('Stat')
plt.title('Random Forest Statistic Importance')
plt.show()
```



In [ ]:

```
```