```python
#!/usr/bin/env python
# coding: utf-8

# In[680]:


import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
get_ipython().run_line_magic('matplotlib', 'inline')


# In[681]:


df = pd.read_csv('collegefootballbowl.csv')


# In[682]:


df


# In[683]:


df.shape


# In[684]:


df.isnull()


# In[685]:


sns.heatmap(df.isnull(), yticklabels=False, cbar=True, cmap='viridis')


# In[686]:


df.head()


# In[687]:


df.drop('winner_rank', axis=1, inplace=True)


# In[688]:


df.drop('loser_rank', axis=1, inplace=True)


# In[689]:


df.drop('sponsor', axis=1, inplace=True)


# In[690]:


df.shape


# In[691]:


sns.heatmap(df.isnull(), yticklabels=False, cbar=True, cmap='viridis')


# In[692]:


df.dropna(inplace=True)
sns.heatmap(df.isnull(), yticklabels=False, cbar=True, cmap='viridis')
day = pd.get_dummies(df['day'], drop_first=True)


# In[693]:


winner_tie = pd.get_dummies(df['winner_tie'], drop_first=True)


# In[694]:
```

```python
loser_tie = pd.get_dummies(df['loser_tie'], drop_first=True)


# In[695]:


mvp = pd.get_dummies(df['mvp'], drop_first=True)


# In[696]:


bowl_name = pd.get_dummies(df['bowl_name'], drop_first=True)


# In[697]:


df = pd.concat([df, day, winner_tie], axis=1)


# In[698]:


df.iloc[:,162] #Wyoming last winner_tie col


# In[699]:


df = pd.concat([df, loser_tie], axis=1)
df


# In[700]:


df.iloc[:,315] #Wyoming last loser_tie col


# In[701]:


df = pd.concat([df, mvp, bowl_name], axis=1)


# In[702]:


df.drop('day',axis=1,inplace=True)
df.drop('winner_tie',axis=1,inplace=True)
df.drop('loser_tie',axis=1,inplace=True)
df.drop('mvp',axis=1,inplace=True)
df.drop('bowl_name',axis=1,inplace=True)
sns.heatmap(df.isnull(), yticklabels=False, cbar=True, cmap='viridis')


# In[703]:


df.isnull()


# In[704]:


df.dropna().shape


# In[705]:


df.shape


# In[706]:


df


# In[707]:


import datetime as dt
import time
def convDate(cols):
    dateComponents = cols[2].split('/')
    month = ""
    for v1 in dateComponents[0]:
        if v1 != "0":
            month += v1
            print(month)
    date = ""
    for v2 in dateComponents[1]:
        if v2 != "0":
            date += v2
            print(date)

    date_time = dt.datetime(int(dateComponents[2]), int(month), int(date))
    print(date_time)
    return time.mktime(date_time.timetuple())


df['date'] = df.apply(convDate, axis=1)


# In[708]:


df


# In[709]:


df.dropna().shape


# In[710]:


df.shape


# In[711]:


sns.heatmap(df.isnull(),cmap='viridis', yticklabels=False, cbar=True)


# In[712]:


from sklearn.model_selection import train_test_split


# In[713]:


df


# In[714]:


df.columns


# In[715]:


print(df.columns)


# In[716]:


def homeWin(cols):
    winPtn = cols[3]
    losePtn = cols[4]
    if winPtn > losePtn + 10:
        return 1
    else:
        return 0


df['landslide'] = df.apply(homeWin, axis=1)


# In[717]:


df


# In[718]:


x = df.drop('landslide', axis=1)
y = df['landslide']


# In[719]:


x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3, random_state=101)
```

```
# In[720]:


from sklearn.linear_model import LogisticRegression


# In[721]:


logmodel = LogisticRegression(max_iter=10000)


# In[722]:


logmodel.fit(x_train, y_train)


# In[723]:


predictions = logmodel.predict(x_test)


# In[724]:


predictions


# In[725]:


print(confusion_matrix(y_test, predictions))
print("\n")
print(classification_report(y_test, predictions))


# In[726]:


df.columns[12:158]

team_wins = []

for j in range(146):
    team_win = pd.DataFrame(columns=df.columns)
    for i in range(1355):
        if df.iloc[i, j+12] == 1:
                team_win.loc[len(team_win.index)] = df.iloc[i]
    print(team_win)
    print("\n")
    team_wins.append(team_win)

team_wins[1].iloc[:,13]
len(team_wins)

team_predictions_winner = []
for i in range(146):
    team_predictions_winner.append(logmodel.predict(team_wins[i].drop('landslide', axis=1)))

team_loses = []

for j in range(146):
    team_lose = pd.DataFrame(columns=df.columns)
    for i in range(1355):
        if df.iloc[i, j+158] == 1:
                team_lose.loc[len(team_lose.index)] = df.iloc[i]
    print(team_lose)
    print("\n")
    team_loses.append(team_lose)


team_predictions_loser = []
for i in range(146):
    team_predictions_loser.append(logmodel.predict(team_loses[i].drop('landslide', axis=1)))


# In[727]:


df.iloc[:,12]


# In[728]:


team_predictions_winner


# In[729]:


def calc_landslide_rate(team_predictions):
    landslide_rate = []
    for i in range(len(team_predictions)):
        avg = 0
        for j in range(len(team_predictions[i])):
            avg += team_predictions[i][j]
        avg /= len(team_predictions[i])
        landslide_rate.append(avg)
    return landslide_rate

landslide_rate_win = calc_landslide_rate(team_predictions_winner)
landslide_rate_lose = calc_landslide_rate(team_predictions_loser)


# In[730]:


team_wins[3]


# In[731]:


len(landslide_rate_lose)


# In[732]:


print(landslide_rate_win, landslide_rate_lose)


# In[733]:


landslideLogDf = pd.DataFrame({'team_name': winner_tie.columns, 'prob_landslide_winning': landslide_rate_win,
                               'prob_landslide_losing': landslide_rate_lose},
                             columns=['team_name','prob_landslide_winning', 'prob_landslide_losing'])


# In[734]:


landslideLogDf


# In[735]:


landslideLogDf.to_csv('FB_predictions/landslide_probability_logistic.csv', index=False)


# In[736]:


from sklearn.neighbors import KNeighborsClassifier


# In[737]:


from sklearn.preprocessing import StandardScaler


# In[738]:


scaler = StandardScaler()


# In[739]:


scaler.fit(df.drop('landslide', axis=1))


# In[740]:


scaled_features = scaler.transform(df.drop('landslide', axis=1))


# In[741]:


scaled_features


# In[742]:


df_feat = pd.DataFrame(scaled_features, columns=df.columns[:-1])


# In[743]:


x_KNN = df_feat
y_KNN = df['landslide']
```

```python
x_train_KNN, x_test_KNN, y_train_KNN, y_test_KNN = train_test_split(x,y, test_size=0.3, random_state=101)


# In[744]:


error_rate = []
for i in range(1,500):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train_KNN, y_train_KNN)
    predictions = knn.predict(x_test_KNN)
    error_rate.append(np.mean(predictions != y_test_KNN))


# In[745]:


predictions = knn.predict(x_test_KNN)


# In[746]:


predictions


# In[747]:


plt.figure(figsize=(10,6))
plt.plot(range(1,500), error_rate, color='green', linestyle='dashed', marker='*', markerfacecolor='red',markersize='5')
plt.title('Error Rate vs K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')


# In[748]:


error_rate.index(min(error_rate))


# In[749]:


scaler2 = StandardScaler()
knn2 = KNeighborsClassifier(n_neighbors=118)
knn2.fit(x_train_KNN, y_train_KNN)


# In[750]:


landslide_predictions_win_knn = []
landslide_predictions_lose_knn = []
for i in range(len(team_wins)):
    landslide_predictions_win_knn.append(knn2.predict(team_wins[i].drop('landslide', axis=1)))
    landslide_predictions_lose_knn.append(knn2.predict(team_loses[i].drop('landslide', axis=1)))




# In[751]:


print(classification_report(team_wins[0]['landslide'], landslide_predictions_win_knn[0]))


# In[752]:


print(confusion_matrix(team_wins[0]['landslide'], landslide_predictions_win_knn[0]))


# In[753]:


landslide_rate_win_knn = calc_landslide_rate(landslide_predictions_win_knn)
landslide_rate_lose_knn = calc_landslide_rate(landslide_predictions_lose_knn)


# In[754]:


landslideKnnDf = pd.DataFrame({'team_name': winner_tie.columns, 'prob_landslide_winning': landslide_rate_win_knn,
                               'prob_landslide_losing': landslide_rate_lose_knn},
                             columns=['team_name','prob_landslide_winning', 'prob_landslide_losing'])


# In[755]:


landslideKnnDf


# In[756]:
```

```
landslideKnnDf.to_csv('FB_predictions/landslide_probability_knn.csv', index=False)


# In[757]:


from sklearn.ensemble import RandomForestClassifier


# In[758]:


rfc = RandomForestClassifier(n_estimators=1000)


# In[759]:


rfc.fit(x_train, y_train)


# In[760]:


landslide_predictions_win_rfc = []
landslide_predictions_lose_rfc = []
for i in range(len(team_wins)):
    landslide_predictions_win_rfc.append(rfc.predict(team_wins[i].drop('landslide', axis=1)))
    landslide_predictions_lose_rfc.append(rfc.predict(team_loses[i].drop('landslide', axis=1)))




# In[761]:


landslide_rate_win_rfc = calc_landslide_rate(landslide_predictions_win_rfc)
landslide_rate_lose_rfc = calc_landslide_rate(landslide_predictions_lose_rfc)


# In[762]:


landslide_rate_win_rfc


# In[763]:


landslideRFCDf = pd.DataFrame({'team_name': winner_tie.columns, 'prob_landslide_winning': landslide_rate_win_rfc,
                               'prob_landslide_losing': landslide_rate_lose_rfc},
                             columns=['team_name','prob_landslide_winning', 'prob_landslide_losing'])


# In[764]:


landslideKnnDf


# In[765]:


landslideKnnDf.to_csv('FB_predictions/landslide_probability_rfc.csv', index=False)


# In[766]:


from sklearn.svm import SVC


# In[767]:


c = []
gamma = []
for i in range(1,1001,5):
    c.append(0.001*i)
    gamma.append(1/i)
param_grid = {'C': c, 'gamma': gamma}


# In[768]:


param_grid


# In[769]:


from sklearn.model_selection import GridSearchCV
```

```python
# In[770]:


grid = GridSearchCV(SVC(), {'C':[0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001,0.0001]}, verbose=10000, refit=True)


# In[771]:


grid.fit(x_train, y_train)


# In[772]:


grid.best_estimator_


# In[773]:


grid.best_params_


# In[774]:


from sklearn.cluster import KMeans


# In[775]:


kmeans = KMeans(n_clusters=2)


# In[776]:


kmeans.fit(x_train, y_train)


# In[777]:


print(classification_report(y_train, kmeans.labels_))


# In[778]:


landslide_predictions_win_svc = []
landslide_predictions_lose_svc = []
for i in range(len(team_wins)):
    landslide_predictions_win_svc.append(grid.predict(team_wins[i].drop('landslide', axis=1)))
    landslide_predictions_lose_svc.append(grid.predict(team_loses[i].drop('landslide', axis=1)))

landslide_rate_win_svc = calc_landslide_rate(landslide_predictions_win_svc)
landslide_rate_lose_svc = calc_landslide_rate(landslide_predictions_lose_svc)



# In[779]:


landslide_rate_win_svc


# In[780]:


landslide_rate_lose_svc


# In[781]:


landslideSVCDf = pd.DataFrame({'team_name': winner_tie.columns, 'prob_landslide_winning': landslide_rate_win_svc,
                               'prob_landslide_losing': landslide_rate_lose_svc},
                             columns=['team_name','prob_landslide_winning', 'prob_landslide_losing'])


# In[782]:


landslideSVCDf


# In[783]:


landslideSVCDf.to_csv('FB_predictions/landslide_probability_SVC.csv', index=False)


# In[784]:
```

```
team_loses


# In[785]:


team_wins


# In[786]:


landslideKnnDf.rename(columns={
    'prob_landslide_winning':'prob_landslide_winning_knn',
    'prob_landslide_losing': 'prob_landslide_losing_knn'}, inplace=True)
landslideLogDf.rename(columns={
    'prob_landslide_winning':'prob_landslide_winning_log',
    'prob_landslide_losing': 'prob_landslide_losing_log'}, inplace=True)
landslideRFCDf.rename(columns={
    'prob_landslide_winning':'prob_landslide_winning_rfc',
    'prob_landslide_losing': 'prob_landslide_losing_rfc'}, inplace=True)
landslideSVCDf.rename(columns={
    'prob_landslide_winning':'prob_landslide_winning_svc',
    'prob_landslide_losing': 'prob_landslide_losing_svc'}, inplace=True)


# In[787]:


landslideCombined = pd.concat([landslideKnnDf, landslideLogDf.iloc[:, 1:3], landslideRFCDf.iloc[:, 1:3], landslideSVCDf.iloc[:, 1:3]], axis=1)


# In[788]:


DLlandslideDf = pd.read_csv("FB_predictions/DL_football_prediction_winner.csv")

landslideCombined = pd.concat([landslideCombined, DLlandslideDf], axis=1)


# In[789]:


DLlandslideDf


# In[790]:


landslide_win_ori = []
for i in range(len(team_wins)):
    landslide_ori_single = []
    for j in range(len(team_wins[i])):
        landslide_ori_single.append(team_wins[i].loc[j, 'landslide'])
    landslide_win_ori.append(landslide_ori_single)


# In[791]:


landslide_rate_win_ori = calc_landslide_rate(landslide_win_ori)


# In[792]:


landslide_rate_win_ori


# In[793]:


landslide_lose_ori = []
for i in range(len(team_loses)):
    landslide_ori_single = []
    for j in range(len(team_loses[i])):
        landslide_ori_single.append(team_loses[i].loc[j, 'landslide'])
    landslide_lose_ori.append(landslide_ori_single)


# In[794]:


landslide_rate_lose_ori = calc_landslide_rate(landslide_lose_ori)


# In[795]:


len(landslide_rate_win_ori)


# In[796]:
```

```
landslideRateActualDf = pd.DataFrame(columns=['actual_landslide_winning','actual_landslide_losing'],
                                     data={'actual_landslide_winning': landslide_rate_win_ori, 'actual_landslide_losing': landslide_rate_lose_ori})

landslideCombined = pd.concat([landslideCombined, landslideRateActualDf], axis=1)


# In[797]:


landslideRateActualDf


# In[798]:


landslideCombined.rename(columns={'team_name':'Team'}, inplace=True)


# In[799]:


winNumDf = pd.read_csv('FB_predictions/linearMergedPrediction.csv')


# In[800]:


winNumDf


# In[801]:


landslideCombined.to_csv('FB_predictions/landslideRatePredictions.csv', index=False)


# In[802]:


newTeamName1 = []
for i in landslideCombined['Team']:
    newTeamName1.append(i.strip().replace(" ", "").lower())

newTeamName2 = []
for i in winNumDf['Team']:
    newTeamName2.append(i.strip().replace(" ", "").lower())


# In[803]:


newTeamName1


# In[804]:


newTeamName2


# In[805]:


landslideCombined = pd.concat([pd.Series(newTeamName1), landslideCombined], axis=1)
landslideCombined.rename(columns={0:'Team_trim'}, inplace=True)


# In[806]:


landslideCombined


# In[807]:


winNumDf = pd.concat([pd.Series(newTeamName2), winNumDf], axis=1)
winNumDf.rename(columns={0:'Team_trim'}, inplace=True)


# In[808]:


mergedDf = pd.merge(landslideCombined, winNumDf, how='inner', on='Team_trim')


# In[809]:


mergedDf


# In[810]:
```

```
mergedDfFinal = mergedDf.drop('Team_trim', inplace=False, axis=1)


# In[811]:


mergedDfFinal.rename(columns={'Team_x':'Team_Name'}, inplace=True)


# In[812]:


mergedDfFinal.drop('Team_y', axis=1, inplace=True)


# In[813]:


mergedDfFinal = pd.concat([mergedDf['Team_trim'], mergedDfFinal], axis=1)
mergedDfFinal.head(25)


# In[814]:


mergedDfFinal.to_csv('FB_predictions/combined_all_predictions.csv', index=False)


# In[815]:


import math
def calculate_team_perf(cols):
    winning_landslide_rates = 1 + np.mean(np.array([cols[1],cols[3],cols[5],cols[7], cols[9]]).astype(float))
    losing_landslide_rates = 1 + np.mean(np.array([cols[2],cols[4],cols[6],cols[8], cols[10]]).astype(float))
    win_predicted = 1 + cols[13]
    loss_predicted = 1 + cols[15]
    return 1 / (1 + math.exp(-(
        (win_predicted * winning_landslide_rates)/(loss_predicted*losing_landslide_rates)
    )))

teamPerf = mergedDfFinal.iloc[:, 1:].apply(calculate_team_perf, axis=1)


# In[816]:


mergedDfFinal


# In[817]:


teamPerf


# In[818]:


teamPerfDf = pd.DataFrame(data={'Team_Name': mergedDfFinal['Team_Name'], 'Team_trimmed_name': mergedDf['Team_trim'], 'Team_Perf_Indicator':teamPerf
                               }, columns={'Team_Name', 'Team_trimmed_name', 'Team_Perf_Indicator'})


# In[819]:


teamPerfDf


# In[820]:


teamPerfDf = pd.concat([teamPerfDf.iloc[:,-1:], teamPerfDf.iloc[:, 1], teamPerfDf.iloc[:,0]], axis=1)


# In[821]:


teamPerfDf


# In[822]:


teamPerfDfCombined = pd.DataFrame(columns=teamPerfDf.columns)
teamPerfArr = []
rowInitial = teamPerfDf.iloc[0,:]

for i in range(0, teamPerfDf.shape[0]):
    if rowInitial[0] == teamPerfDf.iloc[i,0]:
        teamPerfArr.append(teamPerfDf.iloc[i,2])
    else:
        print(teamPerfArr)
        teamPerfDfCombined.loc[len(teamPerfDfCombined.index)] = [rowInitial[0], rowInitial[1], np.mean(teamPerfArr)]
        teamPerfArr = []
        rowInitial = teamPerfDf.iloc[i,:]
```

```python
# In[823]:

teamPerfDfCombined

# In[824]:

teamPerfDf

# In[825]:

teamPerfDfCombined.to_csv('FB_predictions/combinedPerfIndicators.csv', index=False)

# In[826]:

np.exp(1)

# In[827]:

mergedDfFinal

# In[828]:

pd.get_dummies(mergedDfFinal['Team_trim'], drop_first=True)

# In[829]:

teamPerfDfCombined

# In[830]:

x_train

# In[831]:

y_train.to_csv('FB_predictions/y_train(landslide_rate).csv', index=False)

# In[832]:

x_train.to_csv('FB_predictions/x_train.csv', index=False)

# In[833]:

df.to_csv('FB_predictions/df.csv', index=False)

# In[ ]:




# In[ ]:
```