Model Run Setup

In [1]:
```
# Select folder path based on user input
gender = 'M' #input('Enter gender (W for women, M for men): ')

# Assign the appropriate folder path based on the input
MAIN_DIR = './'
USE_DIR = MAIN_DIR + 'womens/' if gender.upper() == 'W' else MAIN_DIR + 'mer
PRE = 'W' if gender.upper() == 'W' else 'M'
NAME = 'womens' if gender.upper() == 'W' else 'mens'
```

In [2]:
```
import pandas as pd
from datetime import datetime, timedelta
import re
import random
import numpy as np
import pickle
```

Load Data

In [3]:
```
# Set the year to get predictions
season = 2023
```

In [4]:
```
# Load 'Sample' data into variable
sample = pd.read_csv('sample-' + NAME + '.csv')
sample.shape
```

Out[4]:
```
(65703, 5)
```

In [5]:
```
# Load 'Sample' data into variable
games = pd.read_csv('games-' + NAME + '.csv')

# Duplicate Games dataframe to get summary statistics
df1 = games[games['Season'] == season].drop(columns=['Team1']).copy()
df2 = games[games['Season'] == season].drop(columns=['Team0']).copy()

# Rename the TeamID columns in dataframe
df1.rename(columns = {'Team0':'TeamID'}, inplace = True)
df2.rename(columns = {'Team1':'TeamID'}, inplace = True)

# Concatenate duplicate dataframes along axis=0 (i.e., add as a new column)
df3 = pd.concat([df1, df2], ignore_index=True, sort=False)

# Group the dataframe by 'Season' and 'TeamID', and get the mean values
team_means = df3.groupby(['Season', 'TeamID'], as_index=False).mean()

# Zero out outcome
team_means['Outcome'] = None

# Show sample output
```

```
print('team_means:', team_means.shape)
team_means.head()
```

```
team_means: (363, 19)
<ipython-input-5-667f6f2a5198>:16: FutureWarning: The default value of nume
ric_only in DataFrameGroupBy.mean is deprecated. In a future version, numer
ic_only will default to False. Either specify numeric_only or select only c
olumns which should be valid for the function.
  team_means = df3.groupby(['Season', 'TeamID'], as_index=False).mean()
```

Out[5]:

	Season	TeamID	Seed	Site	MOV	FG2M	FG2A	FG3M	FG3A
0	2023	1101	0.0	0.153846	-3.692308	-0.423077	6.923077	0.115385	2.230769
1	2023	1102	0.0	0.156250	-0.125000	-0.250000	-2.562500	2.968750	5.218750
2	2023	1103	0.0	0.419355	5.838710	0.677419	0.645161	1.741935	5.096774
3	2023	1104	0.0	0.264706	13.676471	3.823529	-2.176471	4.529412	10.058824
4	2023	1105	0.0	0.033333	-3.066667	0.633333	1.200000	-1.066667	-3.600000

In [6]:
```
# Create Seeds dataframe
seeding = pd.read_csv(USE_DIR + PRE + 'NCAATourneySeeds.csv')

# Reduce size of Seeds dataframe
seeds = seeding[seeding['Season'] == season].copy()

# Create a regex to pull out number
regex = r'\d+'

# Apply the regular expression to the 'col1' column to extract the numeric v
seeds['Seed'] = seeds['Seed'].apply(lambda x: re.search(regex, x).group())

# Convert the numeric values to integers
seeds['Seed'] = seeds['Seed'].astype(int)

# Set order sequence
order = [0, 2, 1]

# Reorder and rename dataframe
seeds = seeds.iloc[:,order]
seeds.rename(columns = {'Seed':'Slot'}, inplace = True)
```

In [7]:
```
# Merge the 'Sample' and 'Seeds' dataframes on 'Season' and 'TeamID' columns
seeds1 = pd.merge(sample, seeds, how='left', left_on=['Season', 'Team0'], ri
seeds1.drop(columns='TeamID', inplace=True)

# Merge the 'Sample' and 'Seeds' dataframes on 'Season' and 'TeamID' columns
seeds2 = pd.merge(seeds1, seeds, how='left', left_on=['Season', 'Team1'], ri
seeds2.drop(columns='TeamID', inplace=True)

# Fill missing values
seeds2['Slot_x'].fillna(17, inplace = True)
seeds2['Slot_y'].fillna(17, inplace = True)
```

```
# Tournament Seeding Consideration (Differential)
seeds2.loc[:, 'Seed'] = np.where(seeds2['Slot_x'] < seeds2['Slot_y'],
                                 seeds2['Slot_x'] - seeds2['Slot_y'],
                                 seeds2['Slot_y'] - seeds2['Slot_x'])
# Convert Seed to int64
seeds2['Seed'] = seeds2['Seed'].astype(int)

# Drop unwanted columns from games dataframe
seeds = seeds2.drop(columns=['Slot_x', 'Slot_y'])

# Add in the remaining columns the model was trained on an set these values
seeds = seeds.assign(Site=0, MOV=0, FG2M=0, FG2A=0, FG3M=0, FG3A=0, FT1M=0,
                     DRB=0, AST=0, TOVR=0, STL=0, BLK=0, PFL=0)
# Set order sequence
order = [0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,

# Reorder columns from games dataframe
seeds = seeds.iloc[:,order].copy()

# Show sample output
print('seeds:', seeds.shape)
print(seeds.columns)
```

```
seeds: (65703, 20)
Index(['Season', 'Team0', 'Team1', 'Seed', 'Site', 'MOV', 'FG2M', 'FG2A',
       'FG3M', 'FG3A', 'FT1M', 'FT1A', 'ORB', 'DRB', 'AST', 'TOVR', 'STL',
       'BLK', 'PFL', 'Outcome'],
      dtype='object')
```

In [8]:
```
# Merge 'Seeds' with 'Team_means' for Team0
df4 = seeds.merge(team_means.add_suffix('_0'), left_on=['Season', 'Team0'],
```

In [9]:
```
# Merge 'Seeds' with 'Team_means' for Team1
df4 = seeds.merge(team_means.add_suffix('_1'), left_on=['Season', 'Team1'],
```

In [10]:
```
# Calculate difference for each column
cols = ['Seed', 'Site', 'MOV', 'FG2M', 'FG2A', 'FG3M', 'FG3A', 'FT1M', 'FT1A
for col in cols:
    df4[col] = df4[col] - df4[col + '_1']
```

In [11]:
```
# Drop redundant columns
df4 = df4.drop(columns=[col + '_1' for col in cols])
df4 = df4.drop(columns=['Season_1', 'TeamID_1'], axis=1)

data = df4.copy()
```

In [12]:
```
# Show sample output
print('merged_means:', data.shape)
print(data.columns)
```

```
merged_means: (65703, 20)
Index(['Season', 'Team0', 'Team1', 'Seed', 'Site', 'MOV', 'FG2M', 'FG2A',
       'FG3M', 'FG3A', 'FT1M', 'FT1A', 'ORB', 'DRB', 'AST', 'TOVR', 'STL',
       'BLK', 'PFL', 'Outcome'],
      dtype='object')
```

Load Model

In [13]:
```
# Load the saved model from a file
loaded_model = pickle.load(open('./models/gnb-model.sav', 'rb'))
```

Run Model on Sample Data

In [14]:
```
# Set X,y variable on sample data
X = data.drop(columns=['Outcome'], axis=1)
y = data['Outcome']

# Make predictions on sample data
y_pred = loaded_model.predict(X)
```

In [15]:
```
# Get probabilities on sample data
y_proba = loaded_model.predict_proba(X)
```

In [16]:
```
len(y_proba)
```

Out[16]:
```
65703
```

Add Probabilities to Sample Dataframe

In [17]:
```
# Create 'ID' column for sample submission
sample['ID'] = sample['Season'].astype(str) + "_" + sample['Team0'].astype(s

# Input the classification results into 'Outcome'
sample['Outcome'] = y_pred

# Create holder column for 'Pred' (probability predictions)
sample['Pred'] = 0.0
```

In [18]:
```
# Iterate over y_pred array and set sample['Outcome'] values
for i in range(len(y_pred)):
    if y_pred[i] == 0:
        sample.at[i, 'Pred'] = y_proba[i][0].round(3)
    else:
        sample.at[i, 'Pred'] = y_proba[i][1].round(3)

# Drop unwanted columns from games dataframe
sample = sample.drop(columns=['Season', 'Team0', 'Team1', 'Seed', 'Outcome']

# Show sample output
print('sample:', sample.shape)
sample.head()
```

```
sample: (65703, 2)
```

Out[18]:

	ID	Pred
0	2023_1101_1102	0.620
1	2023_1101_1103	0.972
2	2023_1101_1104	1.000
3	2023_1101_1105	0.916
4	2023_1101_1106	0.999

In [19]:
```
# Save dataframes as csv files
sample_submission = sample.to_csv('submission-' + NAME + '.csv',index=False)
```

In [20]:
```
sample.Pred.unique()
```

```
Out[20]: array([0.62 , 0.972, 1.   , 0.916, 0.999, 0.975, 0.756, 0.504, 0.845,
                0.979, 0.958, 0.996, 0.814, 0.688, 0.978, 0.919, 0.952, 0.812,
                0.988, 0.969, 0.891, 0.835, 0.844, 0.656, 0.676, 0.94 , 0.869,
                0.875, 0.92 , 0.948, 0.993, 0.99 , 0.59 , 0.834, 0.892, 0.987,
                0.808, 0.992, 0.957, 0.974, 0.781, 0.964, 0.997, 0.915, 0.865,
                0.897, 0.986, 0.868, 0.98 , 0.994, 0.925, 0.629, 0.809, 0.614,
                0.973, 0.55 , 0.743, 0.537, 0.947, 0.591, 0.666, 0.963, 0.649,
                0.894, 0.678, 0.782, 0.953, 0.797, 0.806, 0.991, 0.765, 0.744,
                0.664, 0.938, 0.848, 0.515, 0.878, 0.609, 0.708, 0.586, 0.911,
                0.699, 0.857, 0.551, 0.912, 0.867, 0.566, 0.702, 0.579, 0.705,
                0.989, 0.735, 0.674, 0.715, 0.856, 0.88 , 0.711, 0.832, 0.939,
                0.885, 0.819, 0.871, 0.771, 0.503, 0.529, 0.902, 0.704, 0.732,
                0.623, 0.692, 0.941, 0.583, 0.767, 0.849, 0.946, 0.679, 0.728,
                0.668, 0.763, 0.524, 0.896, 0.684, 0.929, 0.636, 0.841, 0.671,
                0.936, 0.703, 0.961, 0.76 , 0.578, 0.587, 0.758, 0.56 , 0.556,
                0.597, 0.592, 0.855, 0.888, 0.791, 0.626, 0.652, 0.853, 0.83 ,
                0.654, 0.675, 0.935, 0.605, 0.926, 0.928, 0.777, 0.774, 0.646,
                0.803, 0.937, 0.69 , 0.543, 0.833, 0.886, 0.773, 0.872, 0.741,
                0.967, 0.757, 0.663, 0.858, 0.546, 0.548, 0.784, 0.68 , 0.793,
                0.661, 0.695, 0.66 , 0.509, 0.903, 0.512, 0.914, 0.616, 0.779,
                0.821, 0.517, 0.769, 0.805, 0.762, 0.505, 0.854, 0.603, 0.721,
                0.884, 0.906, 0.665, 0.594, 0.954, 0.984, 0.815, 0.879, 0.775,
                0.816, 0.97 , 0.91 , 0.955, 0.933, 0.662, 0.625, 0.982, 0.934,
                0.764, 0.747, 0.985, 0.923, 0.73 , 0.898, 0.638, 0.65 , 0.783,
                0.547, 0.842, 0.602, 0.859, 0.826, 0.683, 0.657, 0.965, 0.648,
                0.77 , 0.905, 0.533, 0.718, 0.836, 0.851, 0.514, 0.506, 0.734,
                0.944, 0.843, 0.921, 0.866, 0.742, 0.552, 0.714, 0.536, 0.87 ,
                0.669, 0.831, 0.557, 0.79 , 0.792, 0.51 , 0.907, 0.511, 0.672,
                0.86 , 0.513, 0.615, 0.549, 0.538, 0.61 , 0.84 , 0.759, 0.604,
                0.593, 0.696, 0.534, 0.507, 0.766, 0.709, 0.698, 0.673, 0.818,
                0.528, 0.624, 0.693, 0.582, 0.523, 0.635, 0.577, 0.677, 0.653,
                0.927, 0.647, 0.667, 0.544, 0.712, 0.755, 0.768, 0.589, 0.588,
                0.516, 0.746, 0.981, 0.628, 0.962, 0.565, 0.761, 0.772, 0.561,
                0.596, 0.852, 0.785, 0.601, 0.837, 0.93 , 0.874, 0.895, 0.796,
                0.731, 0.727, 0.555, 0.78 , 0.822, 0.722, 0.71 , 0.639, 0.825,
                0.502, 0.687, 0.58 , 0.535, 0.691, 0.558, 0.545, 0.811, 0.716,
                0.729, 0.945, 0.813, 0.658, 0.539, 0.611, 0.553, 0.713, 0.745,
                0.581, 0.67 , 0.847, 0.522, 0.778, 0.918, 0.932, 0.697, 0.7  ,
                0.719, 0.508, 0.736, 0.877, 0.733, 0.527, 0.681, 0.634, 0.576,
                0.804, 0.789, 0.883, 0.846, 0.706, 0.694, 0.595, 0.776, 0.754,
                0.6  , 0.627, 0.564, 0.562, 0.873, 0.807, 0.685, 0.89 , 0.554,
                0.64 , 0.655, 0.686, 0.817, 0.726, 0.839, 0.559, 0.723, 0.861,
                0.659, 0.617, 0.54 , 0.682, 0.786, 0.717, 0.701, 0.795, 0.85 ,
                0.612, 0.521, 0.889, 0.913, 0.72 , 0.924, 0.74 , 0.968, 0.526,
                0.633, 0.575, 0.651, 0.613, 0.823, 0.81 , 0.922, 0.599, 0.824,
                0.645, 0.838, 0.737, 0.563, 0.707, 0.904, 0.606, 0.881, 0.909,
                0.518, 0.887, 0.788, 0.753, 0.908, 0.622, 0.641, 0.618, 0.541,
                0.725, 0.52 , 0.724, 0.525, 0.574, 0.829, 0.632, 0.501, 0.644,
                0.876, 0.787, 0.901, 0.598, 0.739, 0.794, 0.519, 0.542, 0.607,
                0.882, 0.621, 0.642, 0.738, 0.532, 0.862, 0.619, 0.752, 0.573,
                0.917, 0.931, 0.585, 0.631, 0.5  , 0.643, 0.942, 0.608, 0.899,
                0.802, 0.584, 0.864, 0.531, 0.748, 0.966, 0.828, 0.572, 0.63 ,
                0.751, 0.53 , 0.863, 0.689, 0.956, 0.801, 0.749, 0.571, 0.9  ,
                0.893, 0.567, 0.75 , 0.827, 0.57 , 0.568, 0.8  , 0.569, 0.951,
                0.995, 0.949, 0.82 , 0.943, 0.799, 0.959, 0.971, 0.983, 0.637,
                0.96 , 0.95 , 0.998])
```