MIS 548 Team Project Final Presentation

By: Mario Lloyd Galvao-Wilson (Data Engineer), Bradley Jones (Business Analyst), and Lucas Sofolo (Marketing)

The Dataset

- Overview and Contents
  - Columns
  - Useful Data
- Our Vision
  - Beneficiaries
  - Possible uses
- Flaws and Errors

```
In [4]: print(nfl.head())
```

```
   nflId  season  teamId   displayName firstName middleName lastName suffix  \
0    182    2008    2120  Xavier Adibi    Xavier    Oyekola    Adibi    NaN
1    182    2009    2120  Xavier Adibi    Xavier    Oyekola    Adibi    NaN
2    182    2010    2120  Xavier Adibi    Xavier    Oyekola    Adibi    NaN
3    182    2011    3000  Xavier Adibi    Xavier    Oyekola    Adibi    NaN
4    182    2012    2100  Xavier Adibi    Xavier    Oyekola    Adibi    NaN

  status positionGroup  ...      gsisId        homeTown collegeId  \
0    ACT            LB  ...  00-0026258  Stillwater, OK      5859
1    ACT            LB  ...  00-0026258  Stillwater, OK      5859
2    ACT            LB  ...  00-0026258  Stillwater, OK      5859
3    ACT            LB  ...  00-0026258  Stillwater, OK      5859
4    ACT            LB  ...  00-0026258  Stillwater, OK      5859

     collegeName  height weight  \
0  Virginia Tech   2-Jun  232.0
1  Virginia Tech   2-Jun  242.0
2  Virginia Tech   2-Jun  242.0
3  Virginia Tech   2-Jun  242.0
4  Virginia Tech   2-Jun  242.0
```

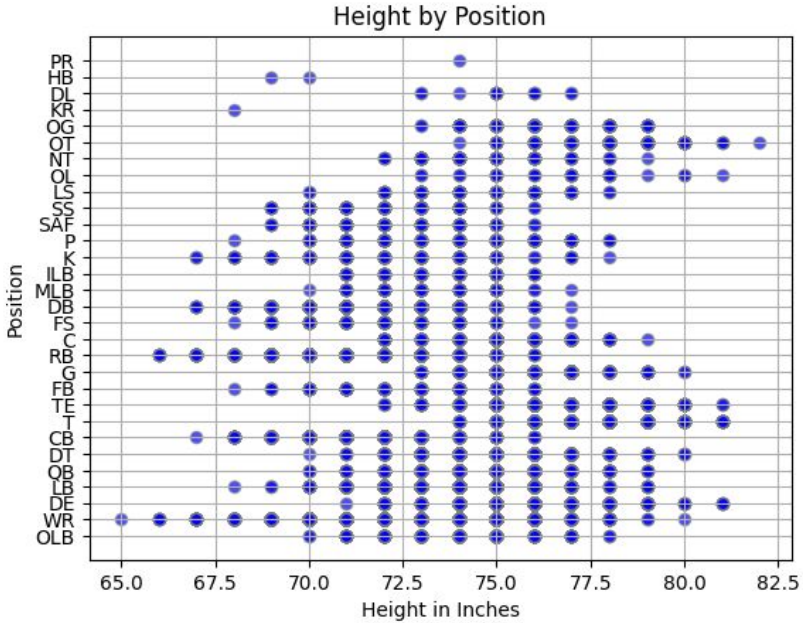
Data Cleaning

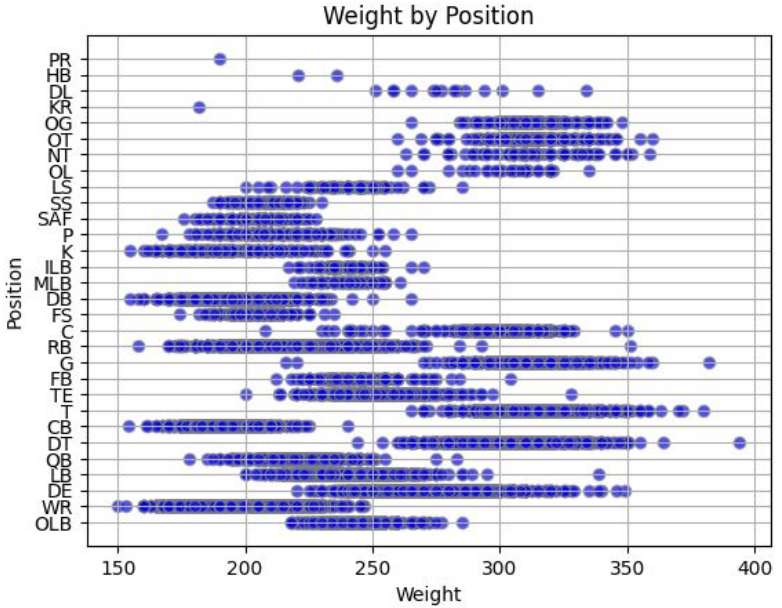
- Use of Pandas
  - Read_csv()
  - head(), info(),
  - unique(),
  - map(),
  - isna(),
  - apply()
- Use of NumPy
  - sum()

In [10]:
```
data = {'Height': ['6\'2', '5\'11', '6\'5', '6\'1', '6\'3', '6\'4', '6\'0', '6\'7', '5\'9',
                   '6\'8', '6\'6', '5\'10', '5\'8', '5\'7', '5\'5', '6\'9', '5\'6', np.nan, '6\'10']}
nfl2 = pd.DataFrame(nfl1)

# Function to convert height to inches
def convert_to_inches(height):
    if pd.isna(height):
        return np.nan

    feet, inches = map(int, height.split('\''))
    total_inches = feet * 12 + inches
    return total_inches

# Create a new column 'Height_Inches'
nfl2['Height_Inches'] = nfl1['Height'].apply(convert_to_inches)
print(nfl2)
```

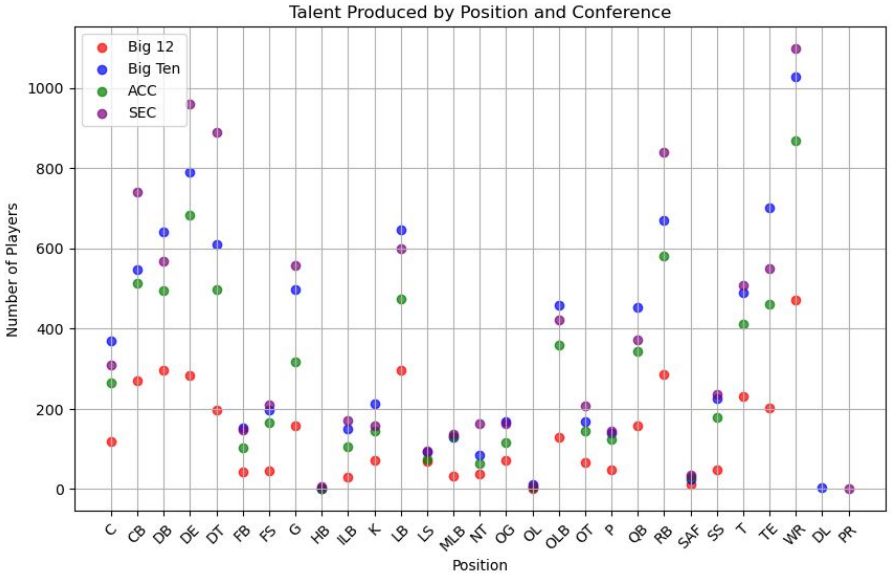
In [87]:
```
agg_functions = {
    'Height_Inches': 'first',
    'weight': 'first',
    'homeTown': 'first',
    'birthDate': 'first',
    'collegeName': 'first',
    'collegeId': 'first',
    'position': 'first',
    'displayName': 'first',
    'season': 'first'
}
the_nfl = nfl2.groupby('nflId').agg(agg_functions).reset_index()

# Print the combined DataFrame
print(the_nfl)
```

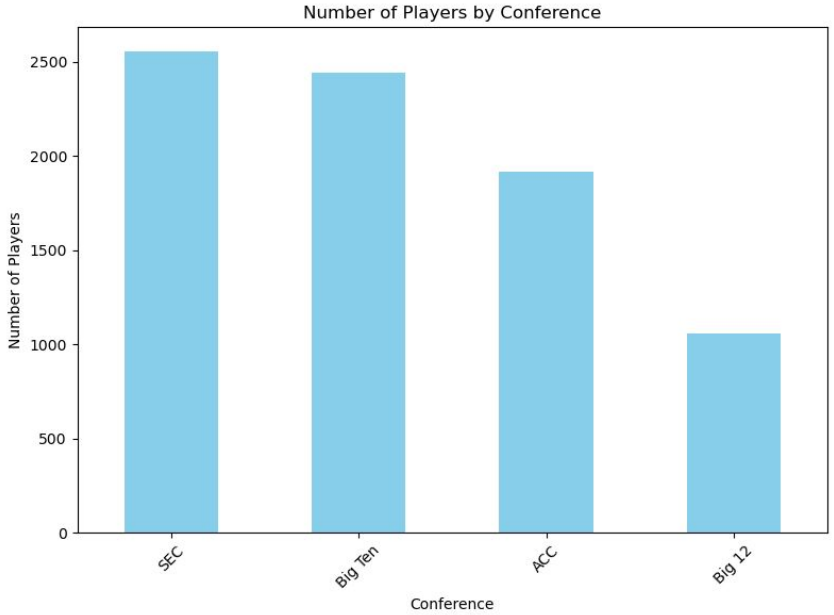
Initial Discovery's and Visuals





Initial Discovery's and Visuals





Machine Learning – Logistic Regression, Classifier Models, and Parameter Tuning

Increasing the iterations of the logistic regression function improved model accuracy by ~8%.

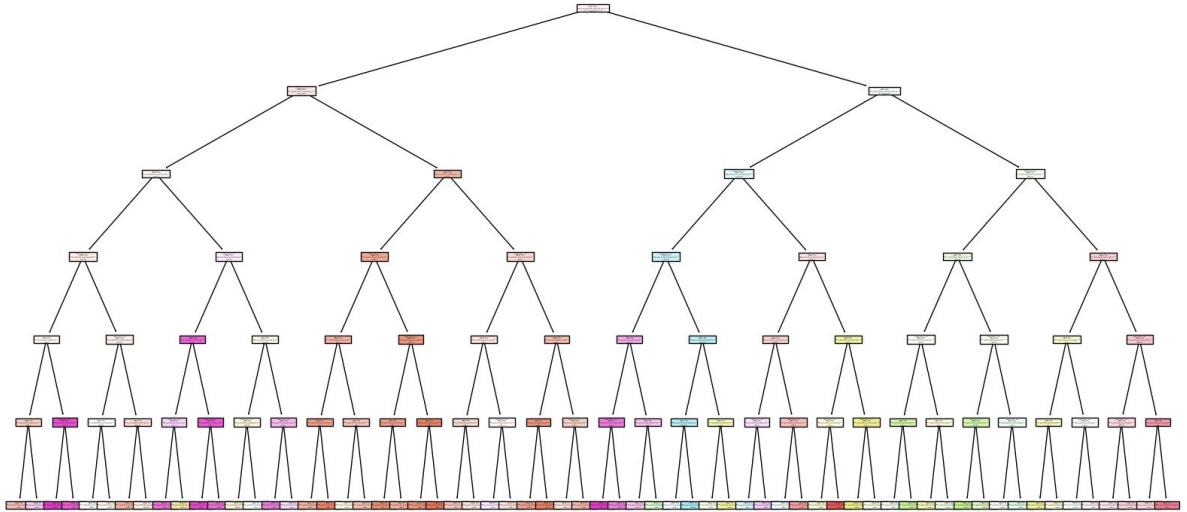
Created a pipeline to scale the data and then train the model; saw ~3% improvement, but an increase in computation time.

Used GridSearchCV to find the best parameters for each model, and use optimal parameters to improve model accuracy.

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [650, 750, 850],
    'max_depth': [4, 6, 8],
    'min_samples_split': [2]
}

grid_search = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

#best params
print(grid_search.best_params_)
```

Machine Learning – Decision Tree Classifier

While a logistic regression model is a good place to start, a decision tree classifier makes much more sense based on the desired use case and the type of our response variable.



Predictive Program

Using the decision tree classifier, we built a program to predict an NFL Player's position based on their height and weight. The model is still not very accurate but with more data and more features to train on, accuracy would likely increase.

```python
from sklearn.tree import DecisionTreeClassifier

# Create a new DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=42, max_depth=6, min_samples_split=2)
dtc.fit(X_train, y_train)

def predict_positiontree(height, weight):
    return dtc.predict(pd.DataFrame([[height, weight]], columns=['Height_Inches', 'weight']))

#test the model
print(predict_positiontree(70, 180))  # Replace with actual height and weight

height = input('Enter height in inches, press ENTER to continue: ')
weight = input('Enter weight in pounds, press ENTER to continue: ')

height = int(height)
weight = int(weight)

print('Based on a height of', height, 'inches and a weight of', weight, 'lbs,', '\n this player should play at the',
      predict_positiontree(height, weight), 'poistion.')

#evaluate the model
y_pred = rfc.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Based on a height of 74 inches and a weight of 175 lbs,
 this player should play at the ['WR'] poistion.
Accuracy: 0.3805386620330148
```

How We Would Market To The NFL & NCAA

- Aid high collegiate and NFL football scouts, recruiters, and coaches in their talent selection process by using past NFL athlete attributes to find potential players at every position
- College: Football scouts and coaches would benefit because the model can help select the right recruits to pursue and spend time, effort, and limited scholarship money on
- NFL: GM's and team scouts would benefit because the insights could help give the highest chance of selecting a player who will be successful in the league

Example Insight

- Offensive tackles (OT) have the largest minimum height of any position
- While our insight is familiar, it could be useful in verifying that short-armed tackles are less likely to be successful in the NFL
- An NFL GM could take this insight when recruiting for tackles, focus only on players taller than 73” inches, and ultimately increase their chance of signing a successful tackle

Players Benefits

- No matter the level, football players will also benefit because they will be put in a position where they have the most potential to succeed
- Experience advantage
- Training knowledge
- Program compatibility

```
    college_counts = position_df['collegeName'].value_counts()
    if not college_counts.empty:
        most_talented_college = college_counts.idxmax()
        talent_at_positions[position]['collegeName'] = most_talented_college
        talent_at_positions[position]['Count'] = college_counts.max()

result_nfl2 = pd.DataFrame.from_dict(talent_at_positions, orient='index')


print("Schools with the most talent at each position:")
print(result_nfl2)
```

```
Schools with the most talent at each position:
          collegeName  Count
OLB               USC     12
LB         Notre Dame     24
WR            Florida     33
DE       Miami (Fla.)     24
QB         Ohio State     11
DT       Miami (Fla.)     20
CB       Miami (Fla.)     19
T                 USC     16
OT            Florida      8
MLB           Clemson      4
ILB           Georgia      7
TE            Stanford    17
NT            Nebraska     3
FB             Alabama     5
G          Notre Dame     16
RB         Notre Dame     22
OL          California     2
C                 USC     11
OG         Ohio State      6
SS                LSU      6
FS             Alabama     6
DB      Virginia Tech     23
K             Georgia      7
```

Next Steps

- With more time, we would expand on other attributes as we did with height, increasing the chance of successful player decisions with each insight
- After success with our original model, we would expand into college player data to help coaches and players at each level
- Include other key performance attributes
- Trustworthy right-hand man for recruiters, coaches, and general managers