ML-based Career Feature

This notebook contains basic analysis and usage of the test model.

1. Introduction

This Machine Learning model provides predictions of career success for candidates based on information such as gender, GPA, age, major and so on. The purpose of the model is to give recruiters a reference to choose employees based on whether the candidate may succeed in a working environment. For any candidate, it gives a result of 1 or 0. 1 means the candidate will succeed and 0 means the candidate will not succeed. The model predicts based on purely objective features.

Test dataset: The test dataset contains 500 candidates data. For each candidate, the information of student ID, gender, age, major, GPA, extracurricular, the number of programming languages the candidate knows, the number of past internships and whether the candidate is a good candidate. The last feature is the feature that we would like to predict using the model. The test dataset has an even distribution for most of the features, as will be explained in section 2. There are close to 50% of the candidates that are good candidates.

Context of use: When recruiters try to interpret the profiles of protential employees, they could refer to the prediction of the model to help them determine if they should hire the candidates.

2. Description of the test data

1. Load model and test dataset

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# matplotlib inline

# Read the data
test_df = pd.read_csv('student_data.csv')
```

In [ ]:
```
# features in test dataset
test_df.head()
```

Out[ ]:

	Student ID	Gender	Age	Major	GPA	Extra Curricular	Num Programming Languages	Num Past Internships	Good Candidate
0	0	F	21	Statistics and Machine Learning	2.83	Sorority	4	1	0
1	1	M	20	Information Systems	2.89	Fraternity	5	3	0
2	2	F	20	Math	2.66	Teaching Assistant	3	1	0
3	3	M	20	Information Systems	2.48	Fraternity	5	0	0
4	4	F	21	Statistics and Machine Learning	3.30	Sorority	2	1	0

In [ ]:
```
# generate descriptive statistics of test dataset
test_df.describe()
```

Out[ ]:

	Student ID	Age	GPA	Num Programming Languages	Num Past Internships	Good Candidate
count	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000
mean	249.500000	20.944000	2.905780	3.04600	2.052000	0.474000
std	144.481833	1.455025	0.839559	1.36073	1.407572	0.499824
min	0.000000	18.000000	0.000000	1.00000	0.000000	0.000000
25%	124.750000	20.000000	2.345000	2.00000	1.000000	0.000000
50%	249.500000	21.000000	2.990000	3.00000	2.000000	0.000000
75%	374.250000	22.000000	3.560000	4.00000	3.000000	1.000000
max	499.000000	25.000000	4.000000	5.00000	4.000000	1.000000

In [ ]:
```
# generate mode of features of test dataset
test_df.mode(numeric_only=False).head(1)
```

Out[ ]:

	Student ID	Gender	Age	Major	GPA	Extra Curricular	Num Programming Languages	Num Past Internships	Good Candidate
0	0	F	21.0	Math	4.0	Student Theatre	2.0	2.0	0.0

From this description, we can see that for most features, they are distributed evenly. The mean value is close to the 50% value. The mode is also close to the mean and median values. Only for the number of programming languages and the GPA, the mode value has great differences from the mean and median value. For GPA, the mode is 4.0, which means there are many more students with high GPA in this dataset. The mode number of programming languages is 2.0, which is much lower than mean 3.046, possibly indicating that this is a feature that might affect the predicting results.

2. Plotting the distribution of the test dataset

To evaluate the fairness of the model, it is important to understand what features it uses to make decisions. In this case, the decision is returning 1 for successful candidates and 0 for unsuccessful candidates. The model makes a decision based on student data, which includes information about age, gender, major, GPA, etc. So to evaluate the fairness of this model, we can examine two features, age and gender, in detail. Age and gender are important when evaluating the fairness of the model because they are attributes that have proven to play a role in hiring decisions. Age is an important feature because we do not want the ML model to only return candidates within a certain age group. If an age bias is present in the sample data, then it will be reflected in the decisions of the ML model. Therefore, it is important to ensure that the ML model uses a fairness strategy that results in the least amount of age bias.

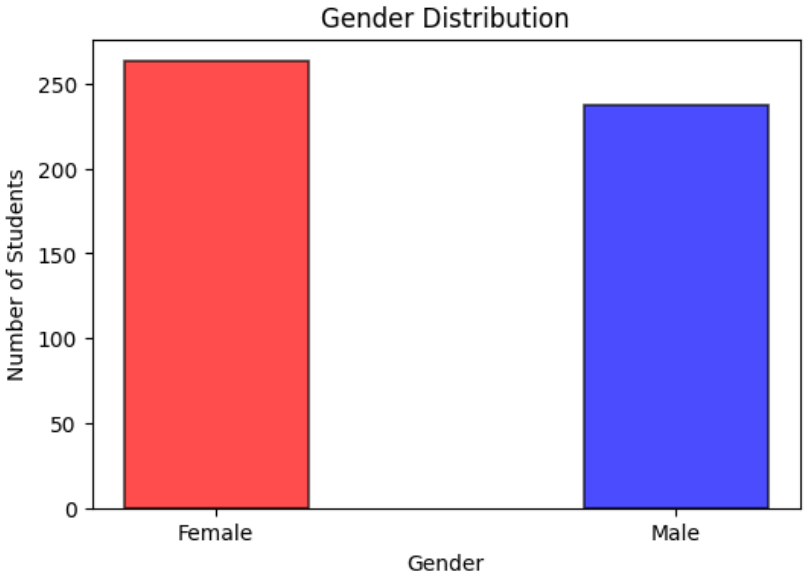
Gender is an important feature that can be used to evaluate the fairness of the ML model. Like age, we do not want the ML model to only return candidates based on gender. So measuring how the ML model reacts to different fairness strategies can be important in improving the ML model, if such a bias does exist. Overall, age and gender are important features we must evaluate to ensure the ML model is fair, as they commonly hold biases. Understanding the impact of these features will improve the ML model.

Gender:

In [ ]:
```
names = ['Female', 'Male']

plt.figure(figsize=(6, 4))
plt.bar(names, test_df.groupby('Gender').size(), width=0.4, color=['red', 'blue'], alpha=0.7, edgecolor='black', li
plt.title('Gender Distribution')
plt.ylabel('Number of Students')
plt.xlabel('Gender')
```

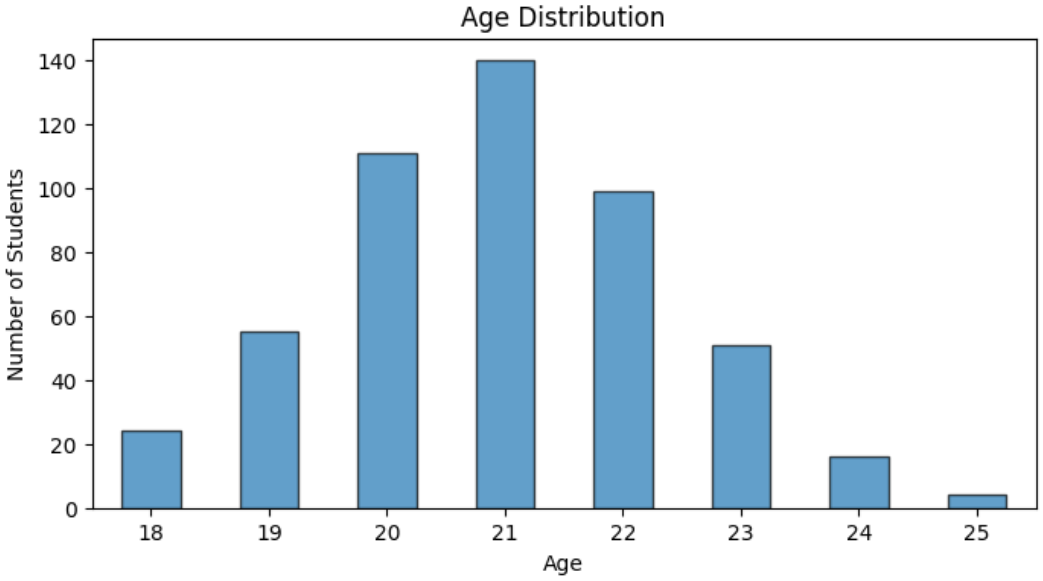
Out[ ]:
```
Text(0.5, 0, 'Gender')
```



Age:

```
In [ ]: test_df.groupby('Age').size().plot(kind='bar', title='Age Distribution', figsize=(8, 4), ylabel='Number of Students
```

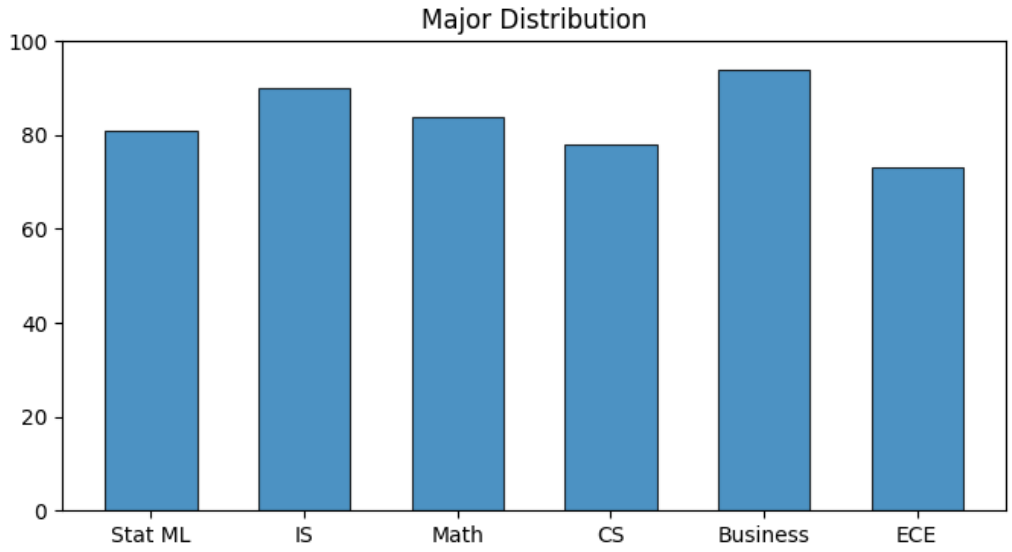
```
Out[ ]: <Axes: title={'center': 'Age Distribution'}, xlabel='Age', ylabel='Number of Students'>
```



Major:

```
In [ ]: names = ['Stat ML', 'IS', 'Math', 'CS', 'Business', 'ECE']
        plt.figure(figsize=(8, 4))
        plt.bar(names, test_df.groupby('Major').size(), width= 0.6, edgecolor='black', linewidth=.8, alpha=0.8)
        plt.yticks(np.arange(0, 110, 20))
        plt.title('Major Distribution')
```

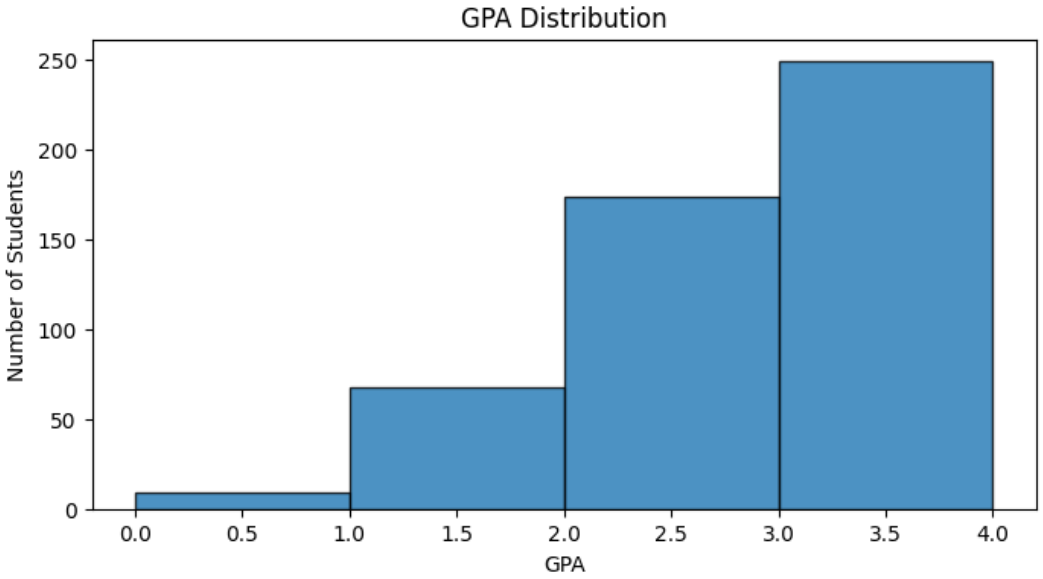
```
Out[ ]: Text(0.5, 1.0, 'Major Distribution')
```



GPA:

In [ ]:
```
plt.figure(figsize=(8, 4))
plt.hist(test_df['GPA'], bins=4, edgecolor='black', linewidth=1, alpha=0.8)
plt.title('GPA Distribution')
plt.xlabel('GPA')
plt.ylabel('Number of Students')
```

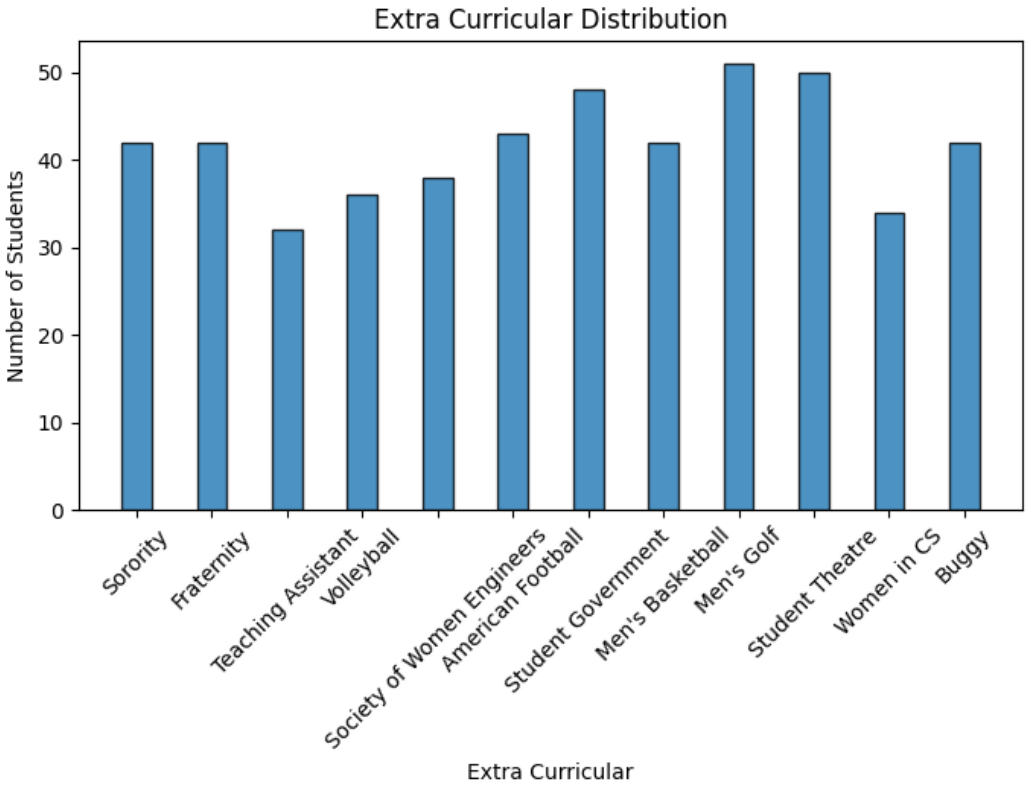
Out[ ]:
```
Text(0, 0.5, 'Number of Students')
```



Extra Curriculars:

In [ ]:
```
names = test_df['Extra Curricular'].unique()
plt.figure(figsize=(8, 4))
plt.bar(names, test_df.groupby('Extra Curricular').size(), width=0.4, edgecolor='black', linewidth=1, alpha=0.8)
plt.title('Extra Curricular Distribution')
plt.ylabel('Number of Students')
plt.xlabel('Extra Curricular')
plt.xticks(rotation=45)
```

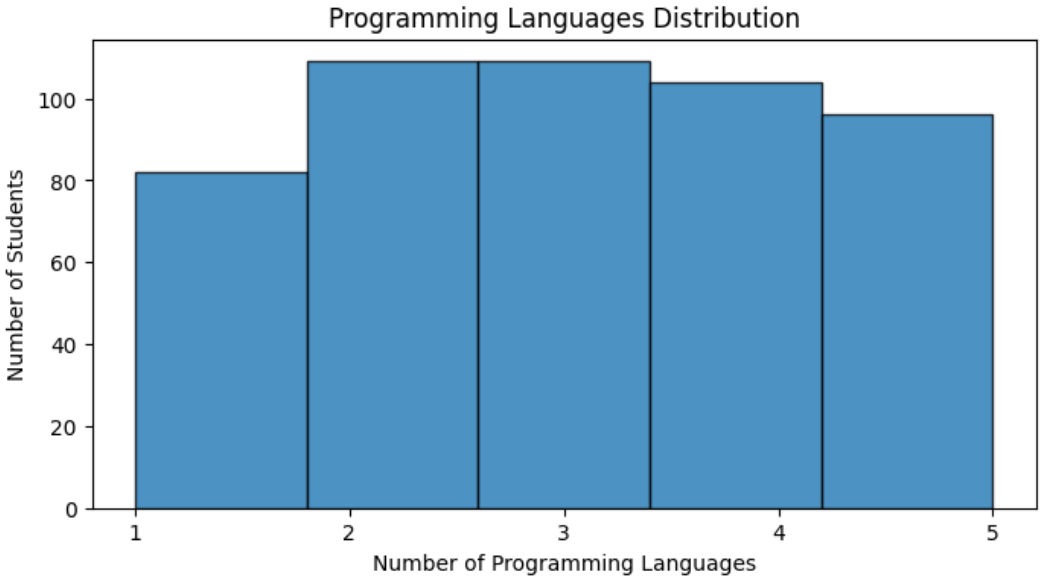
Out[ ]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 [Text(0, 0, 'Sorority'),
  Text(1, 0, 'Fraternity'),
  Text(2, 0, 'Teaching Assistant'),
  Text(3, 0, 'Volleyball'),
  Text(4, 0, 'Society of Women Engineers'),
  Text(5, 0, 'American Football'),
  Text(6, 0, 'Student Government'),
  Text(7, 0, "Men's Basketball"),
  Text(8, 0, "Men's Golf"),
  Text(9, 0, 'Student Theatre'),
  Text(10, 0, 'Women in CS'),
  Text(11, 0, 'Buggy')])
```



Number of Programming Languages:

In [ ]:
```
plt.figure(figsize=(8, 4))
plt.hist(test_df['Num Programming Languages'], bins=5, edgecolor='black', linewidth=1, alpha=0.8)
plt.title('Programming Languages Distribution')
plt.xlabel('Number of Programming Languages')
plt.ylabel('Number of Students')
plt.xticks(np.arange(1, 6, 1))
```

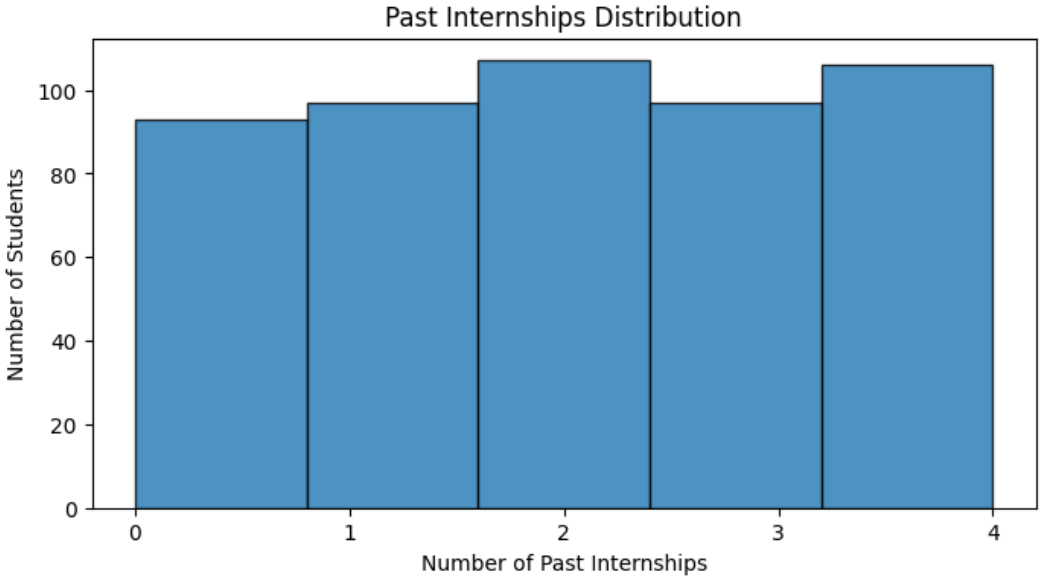
Out[ ]:
```
([<matplotlib.axis.XTick at 0x7ff7e94400d0>,
  <matplotlib.axis.XTick at 0x7ff7e94400a0>,
  <matplotlib.axis.XTick at 0x7ff7e9432cd0>,
  <matplotlib.axis.XTick at 0x7ff7e928ae20>,
  <matplotlib.axis.XTick at 0x7ff7e9297910>],
 [Text(1, 0, '1'),
  Text(2, 0, '2'),
  Text(3, 0, '3'),
  Text(4, 0, '4'),
  Text(5, 0, '5')])
```



Number of Past Internships:

In [ ]:
```
plt.figure(figsize=(8, 4))
plt.hist(test_df['Num Past Internships'], bins=5, edgecolor='black', linewidth=1, alpha=0.8)
plt.title('Past Internships Distribution')
plt.xlabel('Number of Past Internships')
plt.ylabel('Number of Students')
plt.xticks(np.arange(0, 5, 1))
```

Out[ ]:
```
([<matplotlib.axis.XTick at 0x7ff7e94153d0>,
  <matplotlib.axis.XTick at 0x7ff7e924d040>,
  <matplotlib.axis.XTick at 0x7ff7e95817f0>,
  <matplotlib.axis.XTick at 0x7ff7e9205730>,
  <matplotlib.axis.XTick at 0x7ff7e9205cd0>],
 [Text(0, 0, '0'),
  Text(1, 0, '1'),
  Text(2, 0, '2'),
  Text(3, 0, '3'),
  Text(4, 0, '4')])
```



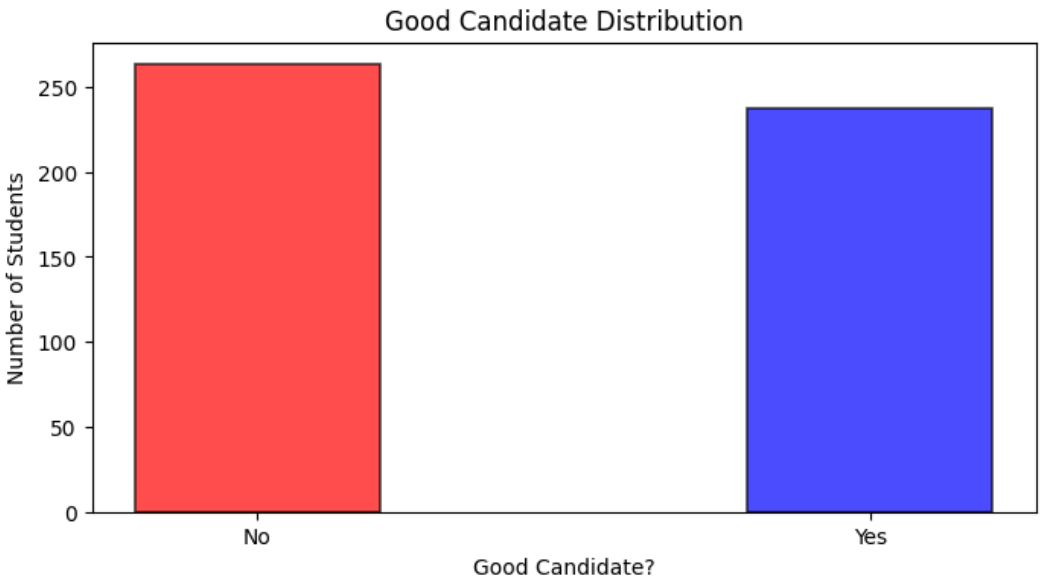
Good Candidates:

In [ ]:
```
names = ['No', 'Yes']

plt.figure(figsize=(8, 4))
plt.bar(names, test_df.groupby('Good Candidate').size(), width=0.4, color=['red', 'blue'], alpha=0.7, edgecolor='bl
```

```
plt.title('Good Candidate Distribution')
plt.ylabel('Number of Students')
plt.xlabel('Good Candidate?')
```

Out[ ]:
```
Text(0.5, 0, 'Good Candidate?')
```



3. Predicting the output of the test dataset using the model

Splitting the data into inputs and the label:

In [ ]:
```
test_df = test_df.drop(['Student ID', 'Gender'], axis=1)
X = test_df.drop(['Good Candidate'], axis=1)
y = test_df['Good Candidate']
X.head()
```

Out[ ]:

	Age	Major	GPA	Extra Curricular	Num Programming Languages	Num Past Internships
0	21	Statistics and Machine Learning	2.83	Sorority	4	1
1	20	Information Systems	2.89	Fraternity	5	3
2	20	Math	2.66	Teaching Assistant	3	1
3	20	Information Systems	2.48	Fraternity	5	0
4	21	Statistics and Machine Learning	3.30	Sorority	2	1

Training model:

In [ ]:
```
X = pd.get_dummies(X)
```

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

# splitting the train and test sets
X_train, X_test, y_train,y_test= train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:
```
clftree = RandomForestClassifier(n_estimators=1000, max_depth=10, max_features=None)
pipeline = Pipeline(steps=[('classifier', clftree)])
pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
```

Exporting model:

In [ ]:
```
import pickle
pickle.dump(pipeline, open('model.pkl','wb'))
```

Exporting test data:

In [ ]:
```
X_test.to_csv('test.csv', index=False)
```

3. Model Performance

Report the accuracy of the model, and the confusion matrix

1. Accuracy:

In [ ]:
```
# Calculate the accuracy of the model
from sklearn.metrics import accuracy_score
print('Accuracy Score', accuracy_score(y_test, y_pred))
```

```
Accuracy Score 0.93
```

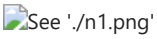
2. Confusion Matrix:

In [ ]:
```
# Show the confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix[0])
print(confusion_matrix[1])
```

```
[46  5]
[ 2 47]
```

The confusion matrix shows the number of predictions that are correct or incorrect in each class. There are in total 46 true negative predictions and 47 true positive predictions, comparing to only 5 false positive and 2 false negative predictions. The overall accuracy is 0.93, which is high for a prediction model. However, in a field like career success prediction, most models do not give accuracy results (From the research that our team has done), thus it is hard for us to compare it to other models.

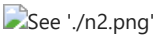
Fairness Evaluation



Strategy 1: Demographic Parity

As shown in theoutput above, the positive rates for all age groups are relatively similar, hovering around the 0.400 to 0.429 range. There is one outlier, being students older than 21 years old, with a positive rate of 0.582. However, that can be explained by the fact that most students older than 21 are seniors about to graduate, meaning these students have the most experience out of everyone else. The important takeaway is that the positive rates for students between 18-21 years old are roughly equal, which means the model is fair in predicting good candidacy based on age when evaluating for demographic parity.

Strategy 2: Equal opportunity



Feature #1: Age

When comparing the positive and negative rates for age groups using an equal opportunity strategy, the model was almost identical across age groups. For 20 year old students, there was a significant contrast between all other age groups. Otherwise, the model had similar rates for age groups 18, 19, 21, and over 21. Since they are roughly equal, we conclude that the model is fair in this fairness strategy.

Feature #2: Gender

When using gender as a feature to evaluate fairness, the model had varying results for male and female students. Using the equal opportunity strategy, male students had a positive rate of 0.914, whereas female students had a positive rate of 0.736. In addition the negative rate for male students was 0.345, and the negative rate was 0.017 for female students. However, since the difference between gender is not as significant compared to the demographic parity strategy, we can conclude that the model is mostly fair when evaluating for equal opportunity.

Strategy 3: Group thresholds

Feature #1: Age

Using group thresholds, we found that the positive rate for students 21 years or younger is 0.821 while the positive rate for students older than 21 years is 0.571. Based on this, we can say that the model has a higher chance of predicting younger students when calculating for group thresholds.

Feature #2: Gender

As demonstrated in the figure above, the positive rate for male students is 0.913. The positive rate for female students is 0.735. Both numbers are relatively high, so gender thresholds are a good predictor for our model.

Results Based on the results of testing different strategies, we found that equal opportunity had the least amount of differences based on age and gender. For example, the positive rate for gender differed by about 0.179 using the equal opportunity strategy. However, using demographic parity resulted in a 0.27 difference between genders. In addition, group thresholds also varied for younger and older students. Since equal opportunity has the most identical rates, we believe it is the most fair strategy to be used.

Recommendation

The model should be used in production because using the equal opportunity strategy, the model can provide correct results. While there is always room for improvement, the model is well-trained. It has a decently high accuracy of 0.832. The model should be used in production because it can help recruiters target specific students. While recruiters shouldn't only use the model to make decisions, the model can be a helpful tool for them.