Mini Project – Programming Laboratory 1

Project by:

Arya Nair- 16010421063

Karan Patel - 16010421072

Problem Statement - To create a machine learning model to predict whether a student will get placement considering the effects of marks in 10th 12th as well as degree marks. Also deploying this ML model on a flask website to create an interface which every ordinary user can easily access.

We use the [Campus recruitment]() dataset available on kaggle. We load the dataset in python using **Pandas** library. In the initial analysis we use **Seaborn** to get the distribution plot to understand the distribution of salary among students.

In [5]:

```
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder

object_cols=['gender','workex','specialisation','status']

# Apply label encoder to each column with categorical data
Q1 = df['hsc_p'].quantile(0.25)
Q3 = df['hsc_p'].quantile(0.75)
IQR = Q3 - Q1
filter = (df['hsc_p'] >= Q1 - 1.5 * IQR) & (df['hsc_p'] <= Q3 + 1.5 *IQR)
placement_filtered=df.loc[filter]

label_encoder = LabelEncoder()
for col in object_cols:
    placement_filtered[col] = label_encoder.fit_transform(placement_filtered[col])
placement_filtered.head()
```

Out[5]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	specialisation	status	salary
0	1	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	0	1	1	270000.0
1	1	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	1	0	1	200000.0
2	1	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	0	0	1	250000.0
3	1	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	0	1	0	NaN
4	1	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	0	0	1	425000.0

then we remove the outliers in the dataset to increase the accuracy. after that we use the label encoder which converts those particular columns into machine readable form. We get it in the 0 or 1 format as there are only two unique values in that column.

In [10]:

```
dummy_hsc_b=pd.get_dummies(placement_filtered['hsc_b'], prefix='dummy')
dummy_ssc_b=pd.get_dummies(placement_filtered['ssc_b'], prefix='dummy')
dummy_hsc_s=pd.get_dummies(placement_filtered['hsc_s'], prefix='dummy')
dummy_degree_t=pd.get_dummies(placement_filtered['degree_t'], prefix='dummy')
placement_coded = pd.concat([placement_filtered,dummy_hsc_s,dummy_degree_t,dummy_hsc_b,dummy_ssc_b],axis=1)
placement_coded.drop(['hsc_b','degree_t','salary','ssc_b','hsc_s'],axis=1, inplace=True)
# placement_coded.head()
# placement_coded.info()
dummy_ssc_b
```

Out[10]:

	dummy_Central	dummy_Others
0	0	1
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
210	0	1
211	0	1
212	0	1
213	0	1
214	1	0

207 rows × 2 columns

After this we converted the columns which had more than two unique values into dummy columns using the pandas method get_dummies(). this converted one single column with multiple unique values into multiple columns with 0s and 1s with respect to the option present in the column.

In [7]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics


X=placement_coded.drop(['status'],axis=1)
y=placement_coded['status']

X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.8,random_state=1)
X_train
```

Out[7]:

	gender	ssc_p	hsc_p	degree_p	workex	specialisation	dummy_Arts	dummy_Commerce	dummy_Science	dummy_Comm&Mgmt	dummy_Others	dummy_Sci&Tech	dun
76	0	66.50	70.40	71.93	0	0	1	0	0	1	0	0	
172	1	73.00	58.00	56.00	0	1	0	1	0	1	0	0	
143	1	77.67	64.89	70.67	0	0	0	1	0	1	0	0	
207	1	83.33	78.00	61.00	1	0	0	1	0	1	0	0	
87	1	59.60	51.00	60.00	0	1	0	0	1	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
138	0	82.00	64.00	73.00	1	0	0	0	1	0	0	1	
142	1	85.00	60.00	73.43	1	0	0	0	1	0	0	1	
75	0	59.00	62.00	77.50	0	1	0	1	0	1	0	0	
145	1	89.40	65.66	71.25	0	1	0	0	1	0	0	1	
38	0	73.00	58.00	66.00	0	1	0	0	1	1	0	0	

165 rows × 16 columns

Then we take the correctly formatted data which is machine readable and separate the data into 2 sections. one of the section we will use to train the model whereas the other section we

will use to check the accuracy of the model.

```
165 rows × 16 columns

In [8]:
model=RandomForestClassifier(n_estimators=100,random_state=1)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
model.score(X_test,y_test)

Out[8]: 0.8095238095238095
```

Then we take the X_train,y_train and train the Random Forest classifier(Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees). After training the model we predict the values using X_test and store the predictions and compare it with y_test to get the accuracy of the model. We first tried logistic regression but then I observed that Random Forest Classifier gave me a better score when using 100 decision trees.

```
with open("model.pkl", "wb") as f:
    pickle.dump(model, f)
```

As the Machine learning model is ready we pickle the same to get a binary file which we can use in our flask web app. By pickling the model we get an already trained model and we don't have to train it again, directly use it to predict the result.

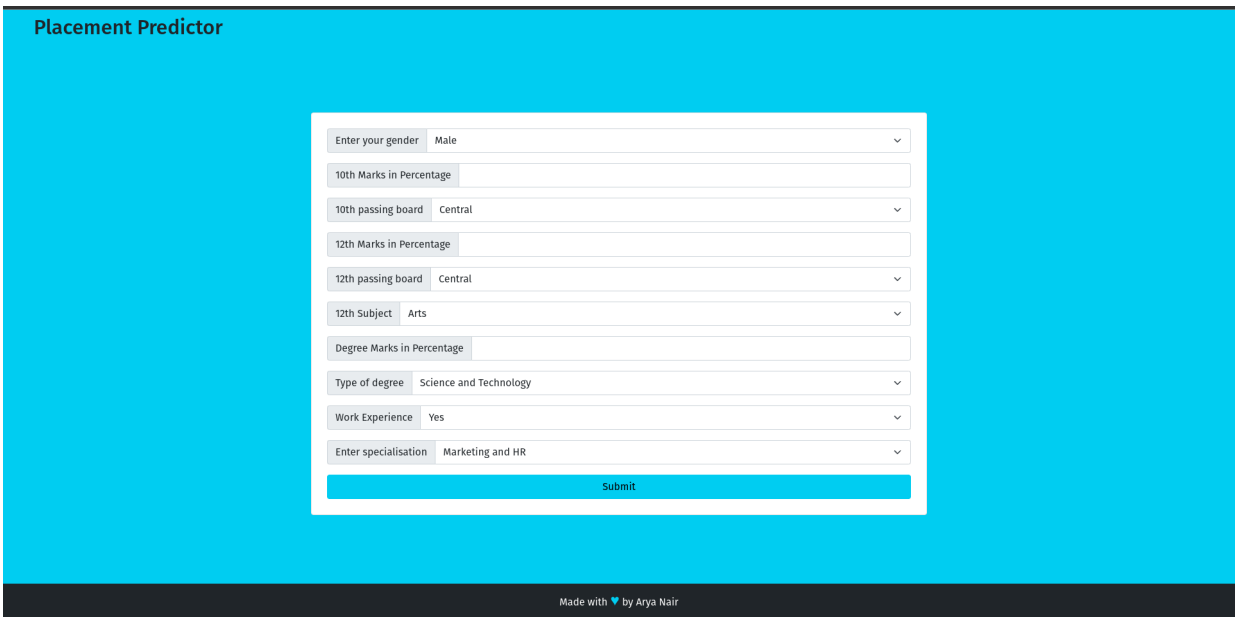
Now we move on to deploying the same on a Flask app.

```
model = pickle.load(open('model.pkl','rb'))

app=Flask(__name__)
```

We load the pickled ML model which we will use later.

```
@app.route('/')
def index():
    return render_template('index.html')
```

This helps me render the index page that we see as soon as we launch the flask app.



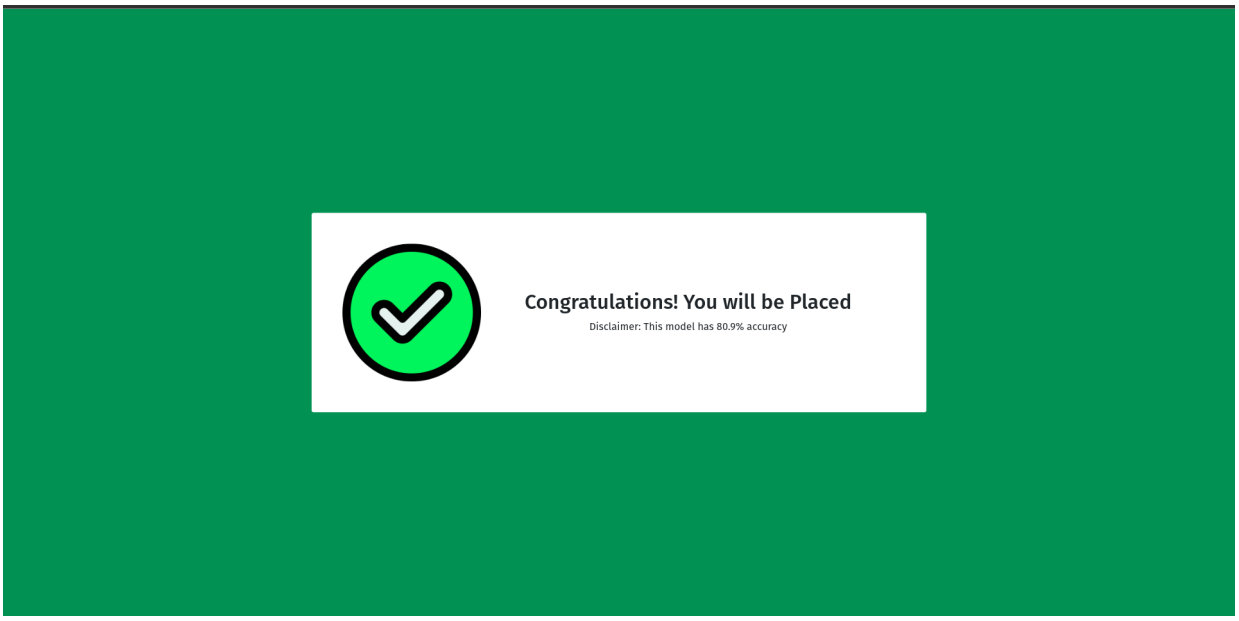
Over here the user enters the relevant information and click submit. When the user clicks submit the data gets sent as a post request to my flask web app.

```python

@app.route('/',methods=['POST'])
def userInput():
    data = request.form.to_dict(flat=False)
    gender=int(data["gender"][0])
    marks10th=int(data["10thMarks"][0])
    marks12th=int(data["12thMarks"][0])
    degreeMarks=int(data["degreeMarks"][0])
    workExp=int(data["workExp"][0])
    specialisation=int(data["specialisation"][0])

    X_test=[gender,marks10th,marks12th,degreeMarks,workExp,specialisation]

    subject12th=data["12thSubject"][0]
    if subject12th=="Science":
        X_test+=[0,0,1]
    elif subject12th=="Arts":
        X_test+=[1,0,0]
    else:
        X_test+=[0,1,0]

    degreeType=data["degreeType"][0]
    if degreeType=="Sci&Tech":
        X_test+=[0,0,1]
    elif degreeType=="Comm&Mgmt":
        X_test+=[1,0,0]
    else:
        X_test+=[0,1,0]

    board12th=data["12thBoard"][0]
    if board12th=="Central":
        X_test+=[1,0]
    else:
        X_test+=[0,1]

    board10th=data["10thBoard"]
    if board10th=="Central":
        X_test+=[1,0]
    else:
        X_test+=[0,1]
    prediction=model.predict([X_test])

    if prediction[0]==1:
        return render_template('placed.html')
    else:
        return render_template('notplaced.html')

```

As soon as we receive the request. we start formatting the data into the format the machine learning model requires. As soon as we get the desired format we pass the data to the ML model.

if we get the prediction as 0 we render the fail template.

and if we get the prediction as 1 we render the success template



To try this project on your own system

README.md

Placement Predictor

Placement Predictor is a web app made using Flask which uses a Machine Learning model to determine whether the user will get placement or not.

Installation

Use the package manager pip to install foobar.

```
virtualenv virt
source virt/bin/activate
cd app
pip install -r requirements.txt
python app.py
```

Usage

```
Open
http://localhost:5000/
to see the website in action
```

Website in Action

For seeing the entire code as well README I would love if you see the Project on github

https://github.com/Arya-A-Nair/PlacementPredictor