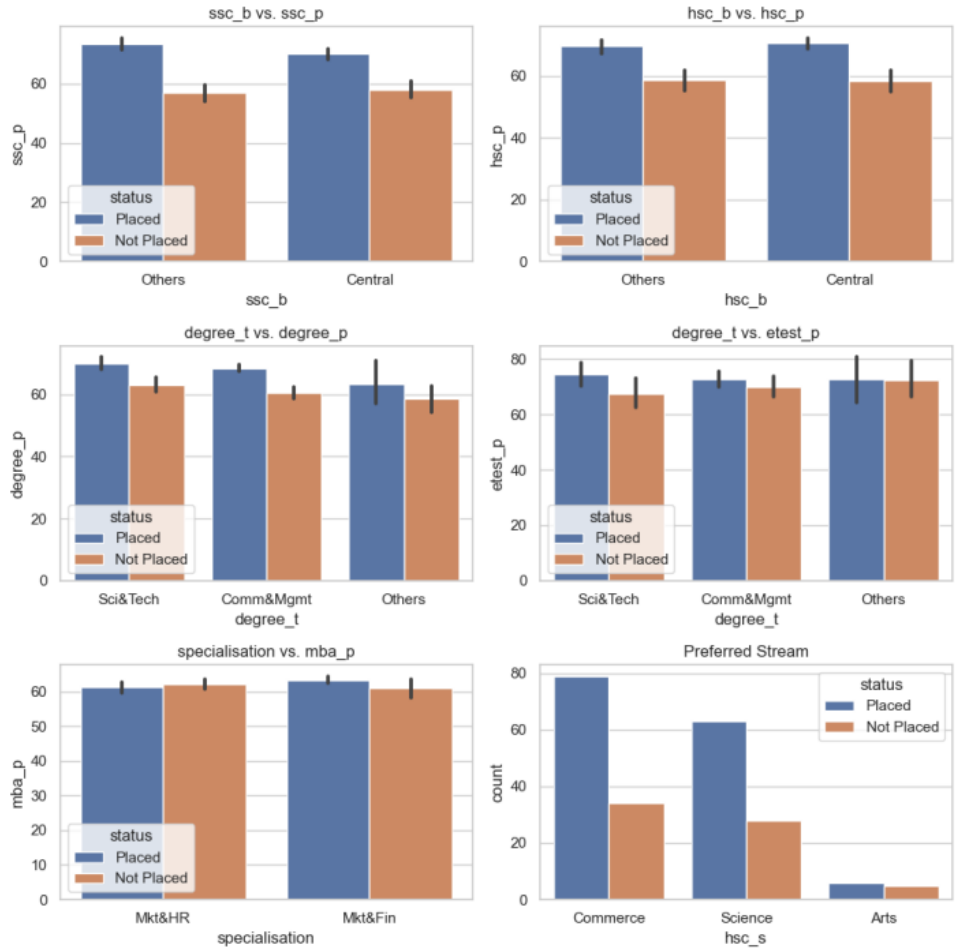
Challenge Title : IBM Hack Challenge 2023

Project ID : SPS_PRO_3625

Project Title : Identifying Patterns and Trends in Campus Placement Data using Machine Learning

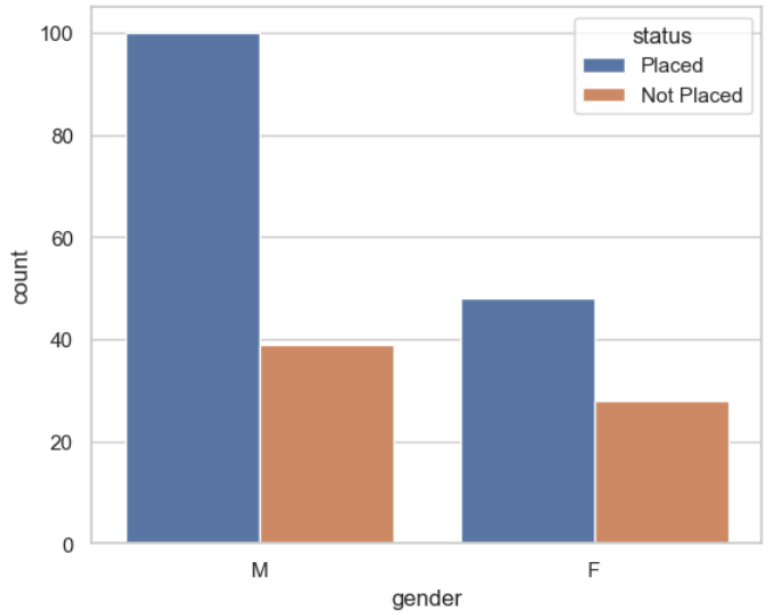
Analyzing the given campus placement data:-

After thorough analysis of the data these insights were gained:

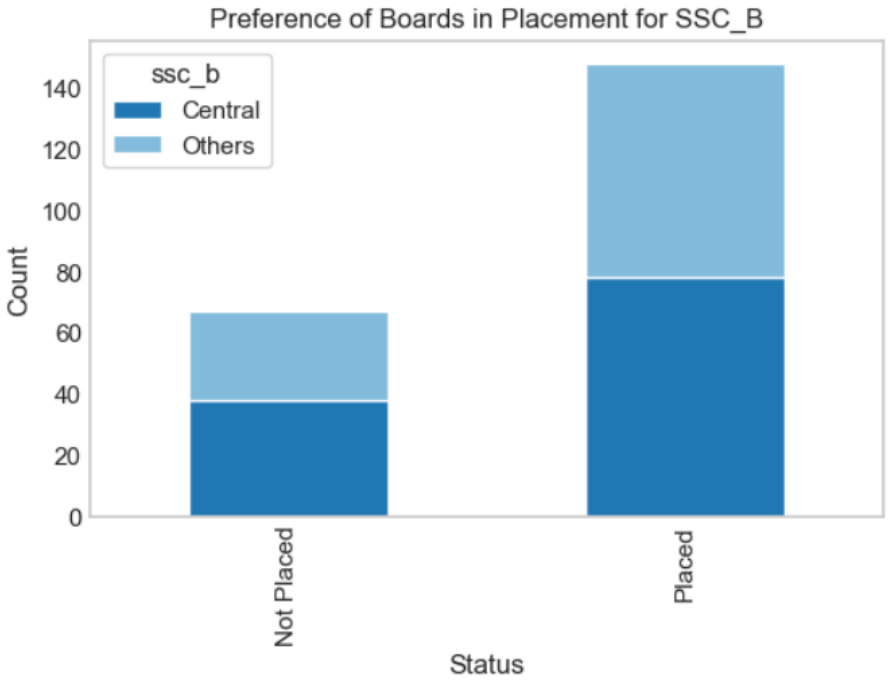


1) Students who did not get a placement were having a percentage less than 60% in their secondary and higher secondary education.

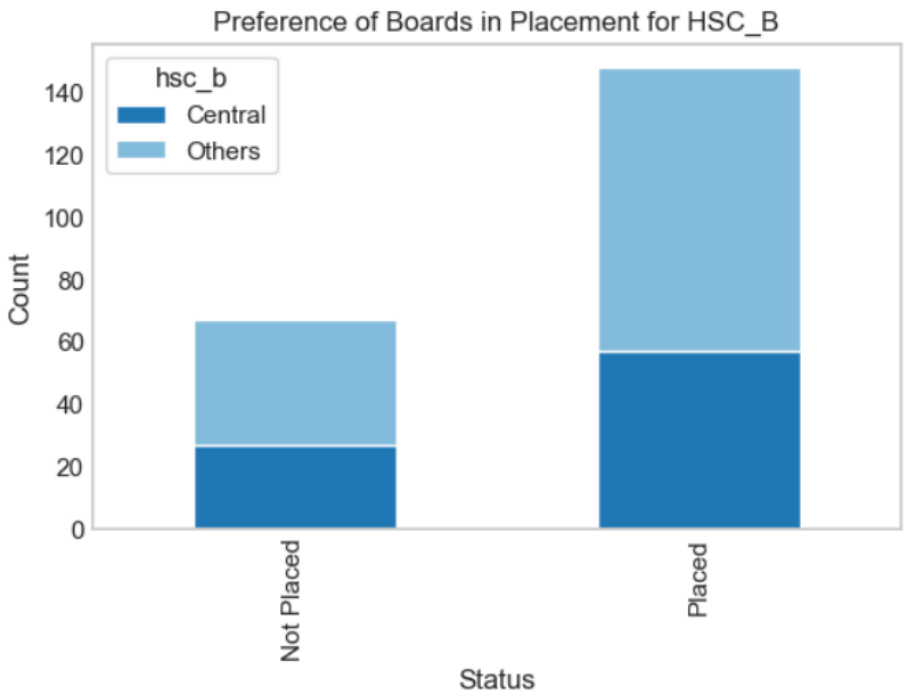
2) The preferred stream in placement process is commerce then science.



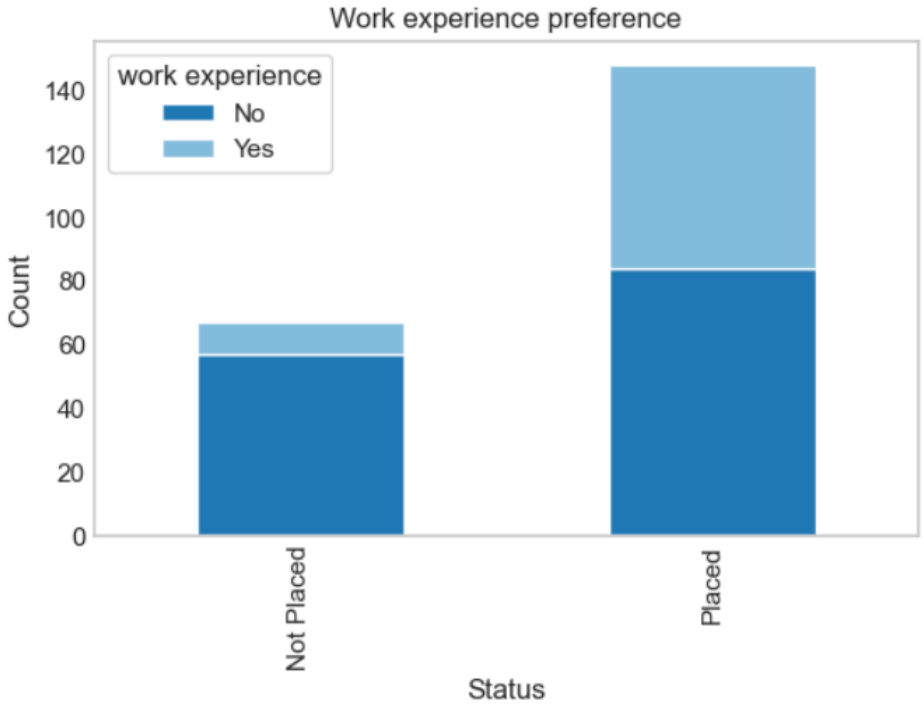
3) It seems like there is a slight gender gap between male and female students but it can be neglected as male students are higher in ratio.



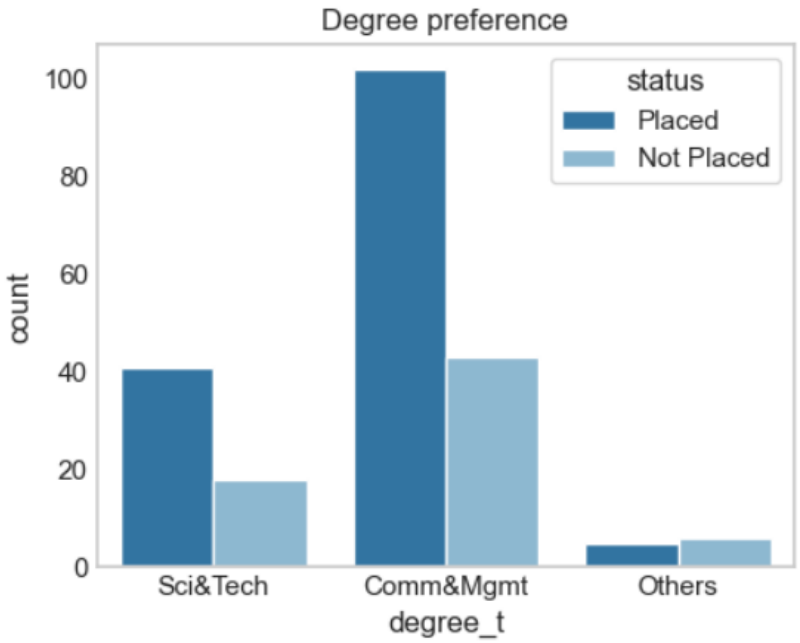
4) There is no such preference between secondary board as per the above plot.



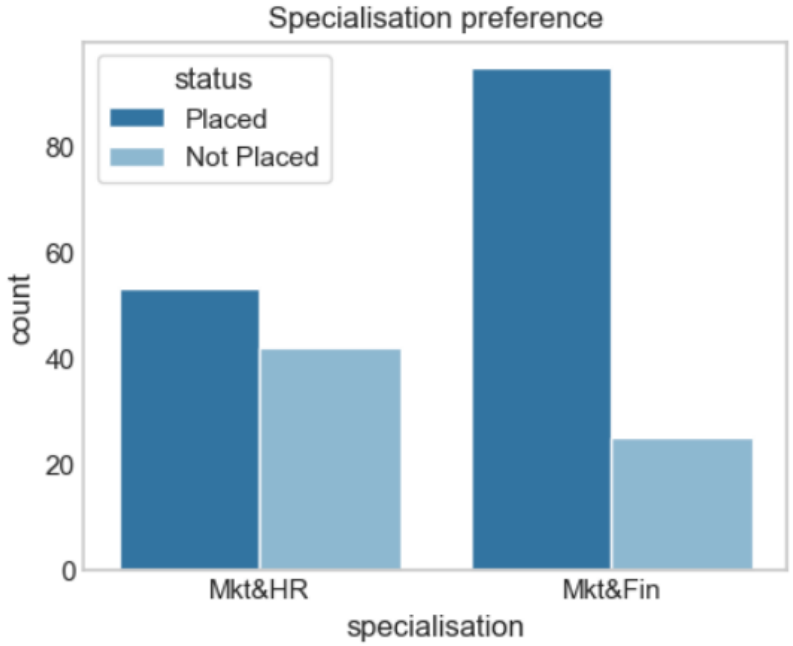
5) The above plot shows that companies are very likely to take students from non-central boards in the Higher Secondary Education of Students.



6) The above plot shows that most of students who didn't get a placement were not having work eperience but also most the students who got placed were not having workexperience. This means academics has a higher hand over work experience.



7) The above plot shows that most preferred degree type is Commerce and Management. Thereafter comes Science and Technology.



8) By the above plot we can see that Marketing and Finance is the most preferred specialisation.

	Model	Score
4	Naive Bayes	93.023256
3	Linear Discriminant Analysis	86.046512
5	Support Vector Machines	86.046512
0	Logistic Regression	83.720930
6	K - Nearest Neighbors	83.720930
2	Random Forest	81.395349
1	Decision Tree	76.744186

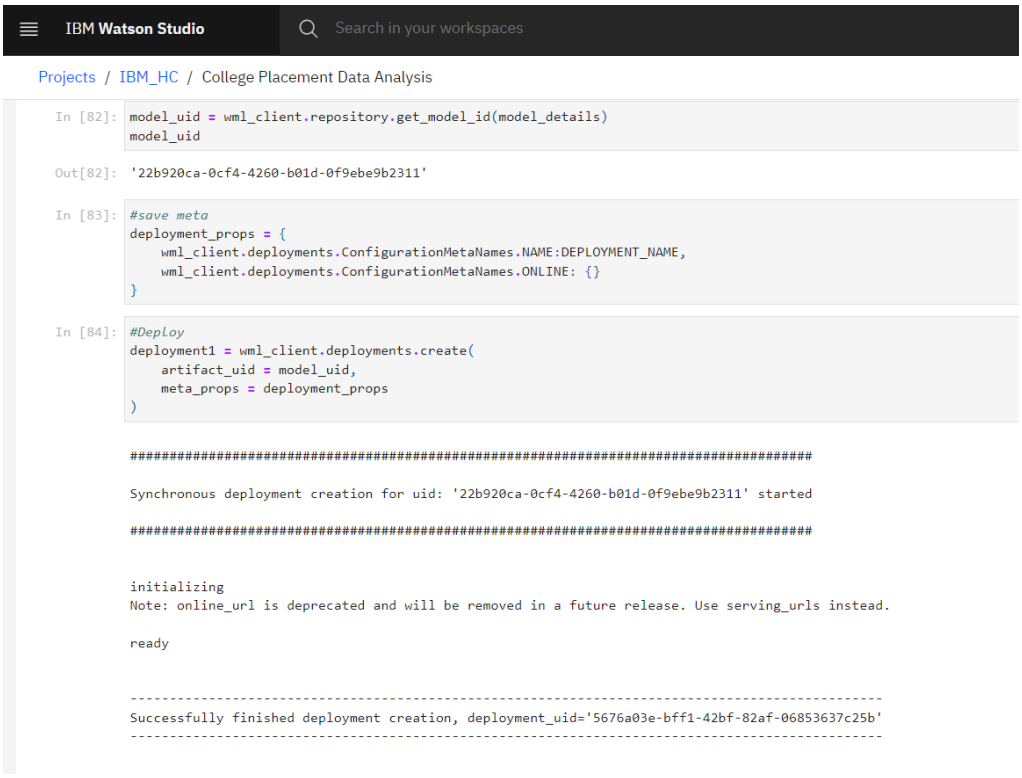
9) The above data shows that Gaussian Naive Bayes works good for the campus placement data with a 93% accuracy.

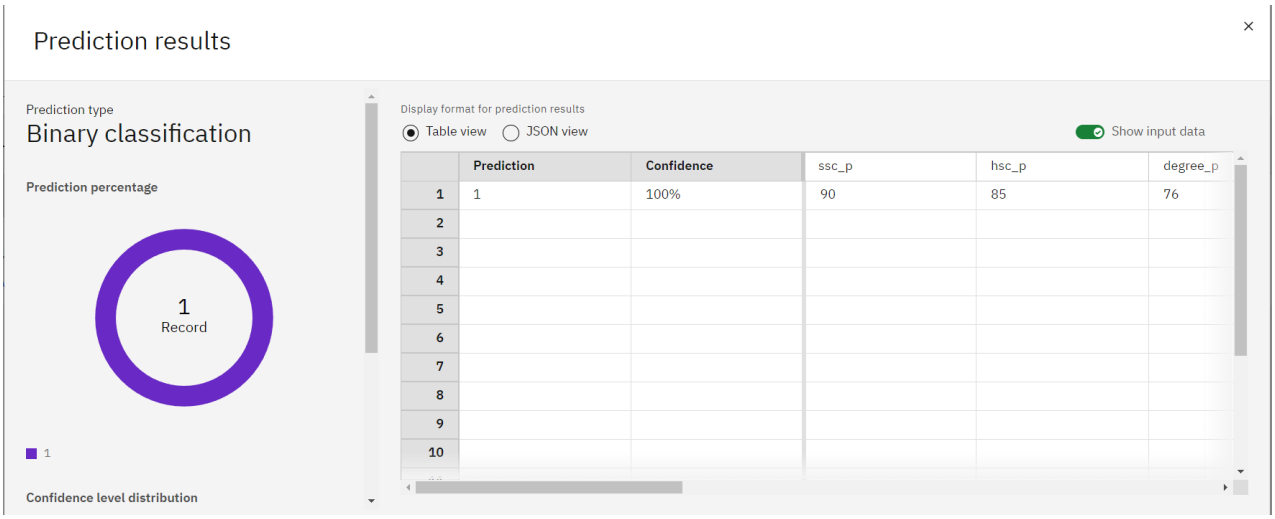
```
Average Accuracy: 81.3953488372093
Standard Deviation of Accuracy: 0.03891441983879421
```

10) The KFold validation tells us that we have a good average accuracy of 81% and standard deviation of 0.03.
Then the model was saved in a pickle file.

Integrating with IBM Watson:-

Created a machine learning instance with IBM Watson Studio to enable a cloud computing environment for prediction. It requires to generate an API key which is used by the flask application to communicate through the model saved in the cloud platfrom.





Making the Flask application:-

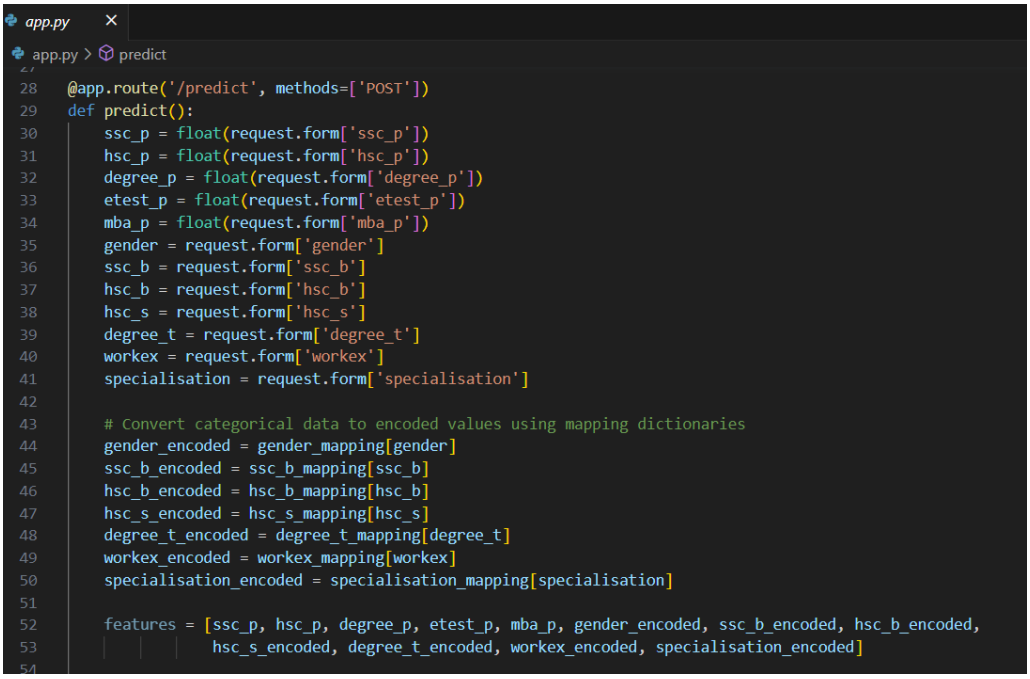
First made a template which will be the home page where the user will be putting the details of the student to do the prediction for him/her.
Then created a python file and imported the flask module. Created statement for related prediction outcomes and mappings for different binary inputs.

app.py

app.py > predict

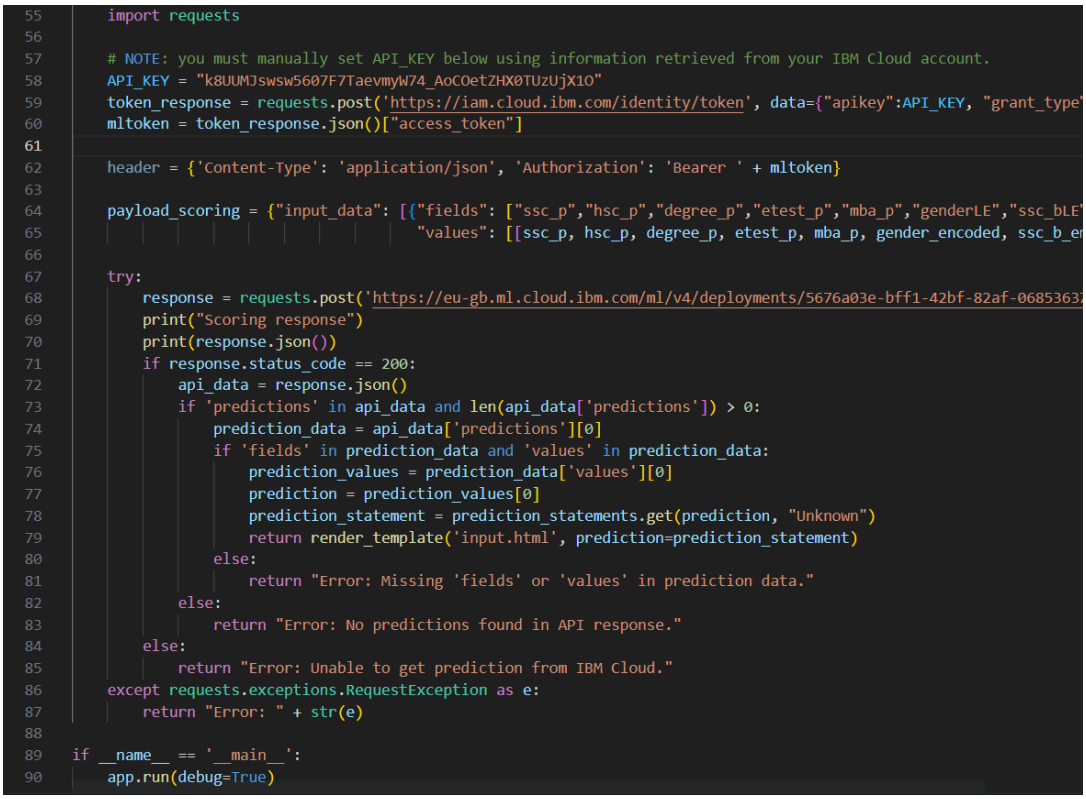
```python
from flask import Flask, render_template, request, jsonify
import requests
import pickle

app = Flask(__name__)

model_filename = 'classification_gnb.pkl'
with open(model_filename, 'rb') as file:
    model = pickle.load(file)

prediction_statements = {
    0: "Sorry, You cannot get a placement.",
    1: "Congartulations! You can get a placement."
}

gender_mapping = {'F': 0, 'M': 1}
ssc_b_mapping = {'Central': 0, 'Others': 1}
hsc_b_mapping = {'Central': 0, 'Others': 1}
hsc_s_mapping = {'Arts': 0, 'Commerce': 1, 'Science': 2}
degree_t_mapping = {'Comm&Mgmt': 0, 'Others': 1, 'Sci&Tech': 2}
workex_mapping = {'No': 0, 'Yes': 1}
specialisation_mapping = {'Mkt&Fin': 0, 'Mkt&HR': 1}

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

The next step was to set all the variables which were requested from the template and decoding the mapped variables.



```python
@app.route('/predict', methods=['POST'])
def predict():
    ssc_p = float(request.form['ssc_p'])
    hsc_p = float(request.form['hsc_p'])
    degree_p = float(request.form['degree_p'])
    etest_p = float(request.form['etest_p'])
    mba_p = float(request.form['mba_p'])
    gender = request.form['gender']
    ssc_b = request.form['ssc_b']
    hsc_b = request.form['hsc_b']
    hsc_s = request.form['hsc_s']
    degree_t = request.form['degree_t']
    workex = request.form['workex']
    specialisation = request.form['specialisation']

    # Convert categorical data to encoded values using mapping dictionaries
    gender_encoded = gender_mapping[gender]
    ssc_b_encoded = ssc_b_mapping[ssc_b]
    hsc_b_encoded = hsc_b_mapping[hsc_b]
    hsc_s_encoded = hsc_s_mapping[hsc_s]
    degree_t_encoded = degree_t_mapping[degree_t]
    workex_encoded = workex_mapping[workex]
    specialisation_encoded = specialisation_mapping[specialisation]

    features = [ssc_p, hsc_p, degree_p, etest_p, mba_p, gender_encoded, ssc_b_encoded, hsc_b_encoded,
                hsc_s_encoded, degree_t_encoded, workex_encoded, specialisation_encoded]
```

Finally imported all the requests snd conneted to the Watson API.



```python
    import requests

    # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
    API_KEY = "k8UUMJswsw5607F7TaevmyW74_AoCOetZHX0TUzUjX1O"
    token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":API_KEY, "grant_type"
    mltoken = token_response.json()["access_token"]

    header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

    payload_scoring = {"input_data": [{"fields": ["ssc_p","hsc_p","degree_p","etest_p","mba_p","genderLE","ssc_bLE"
                                                  "values": [[ssc_p, hsc_p, degree_p, etest_p, mba_p, gender_encoded, ssc_b_e

    try:
        response = requests.post('https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/5676a03e-bff1-42bf-82af-0685363
        print("Scoring response")
        print(response.json())
        if response.status_code == 200:
            api_data = response.json()
            if 'predictions' in api_data and len(api_data['predictions']) > 0:
                prediction_data = api_data['predictions'][0]
                if 'fields' in prediction_data and 'values' in prediction_data:
                    prediction_values = prediction_data['values'][0]
                    prediction = prediction_values[0]
                    prediction_statement = prediction_statements.get(prediction, "Unknown")
                    return render_template('input.html', prediction=prediction_statement)
                else:
                    return "Error: Missing 'fields' or 'values' in prediction data."
            else:
                return "Error: No predictions found in API response."
        else:
            return "Error: Unable to get prediction from IBM Cloud."
    except requests.exceptions.RequestException as e:
        return "Error: " + str(e)

if __name__ == '__main__':
    app.run(debug=True)
```

Testing the application:-

Simply running the app.py file to see the locally deployed application.

Enter the test values and predict the outcome:-