

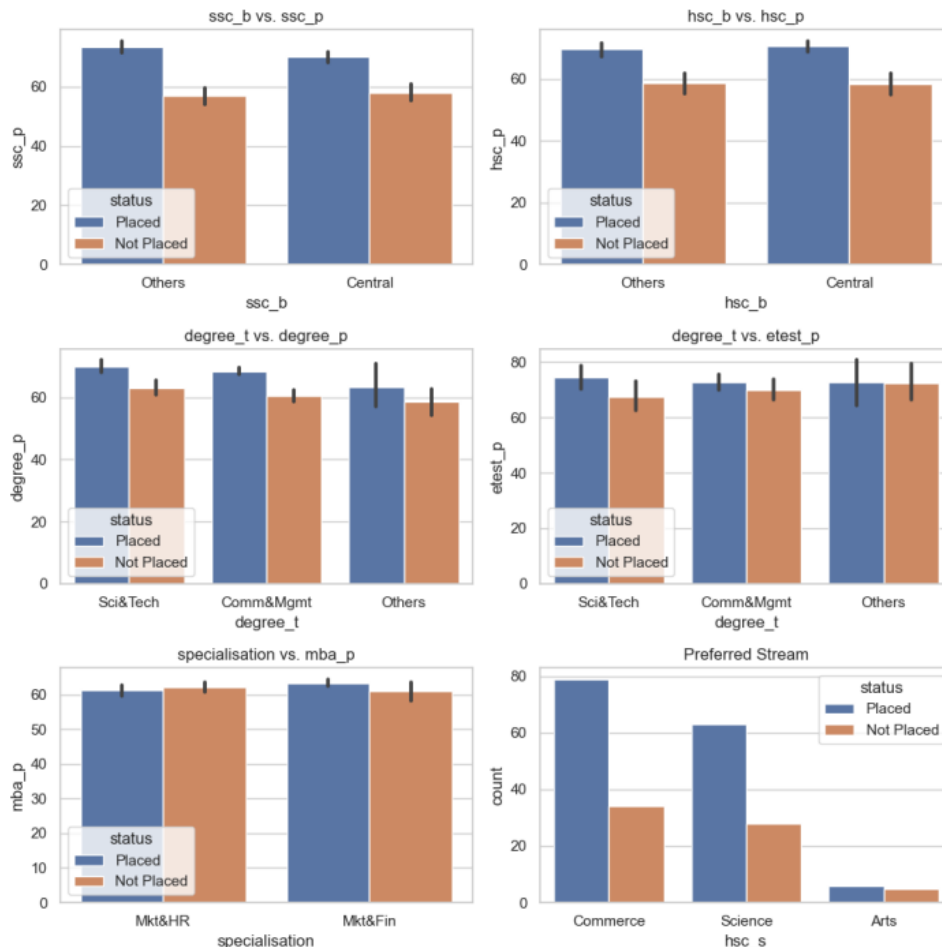
Challenge Title : IBM Hack Challenge 2023

Project ID : SPS_PRO_3625

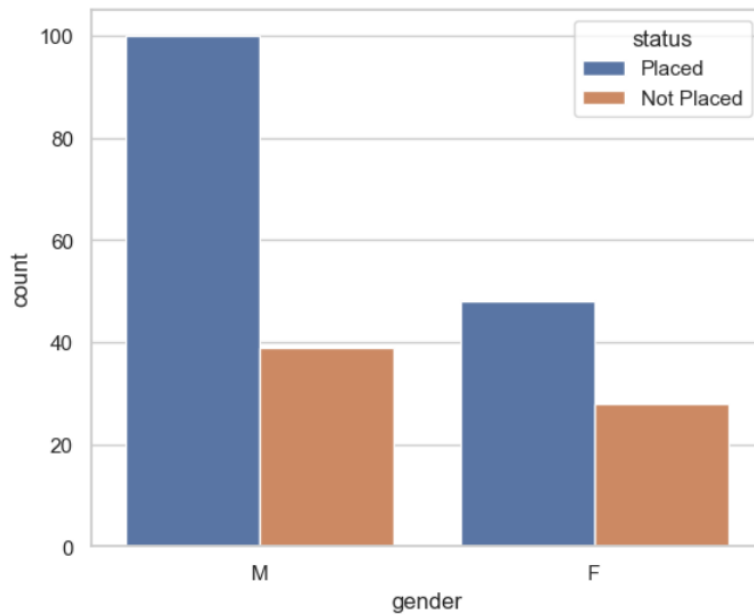
Project Title : Identifying Patterns and Trends in Campus Placement Data using Machine Learning

Analyzing the given campus placement data:-

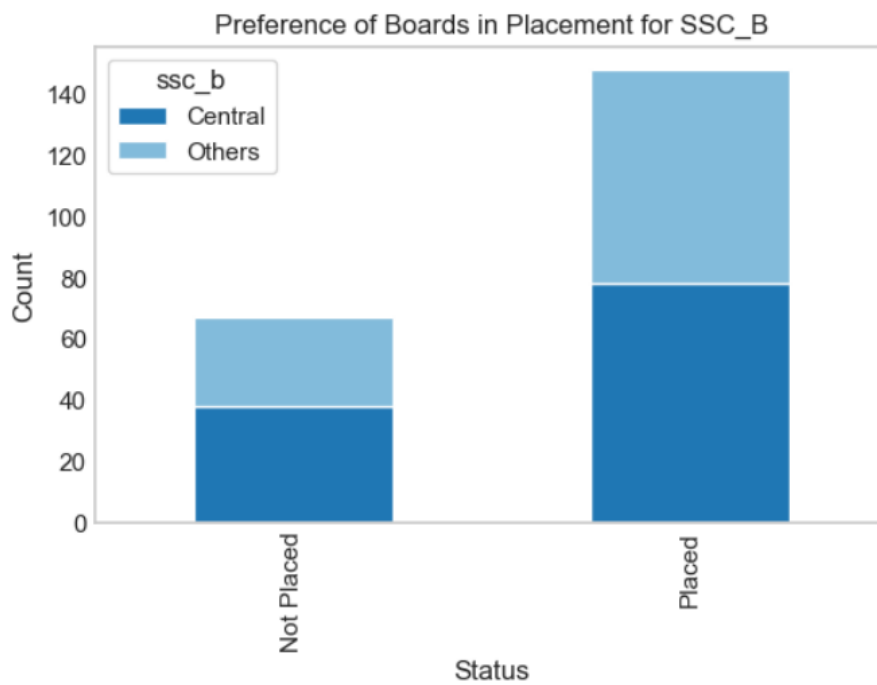
After thorough analysis of the data these insights were gained:



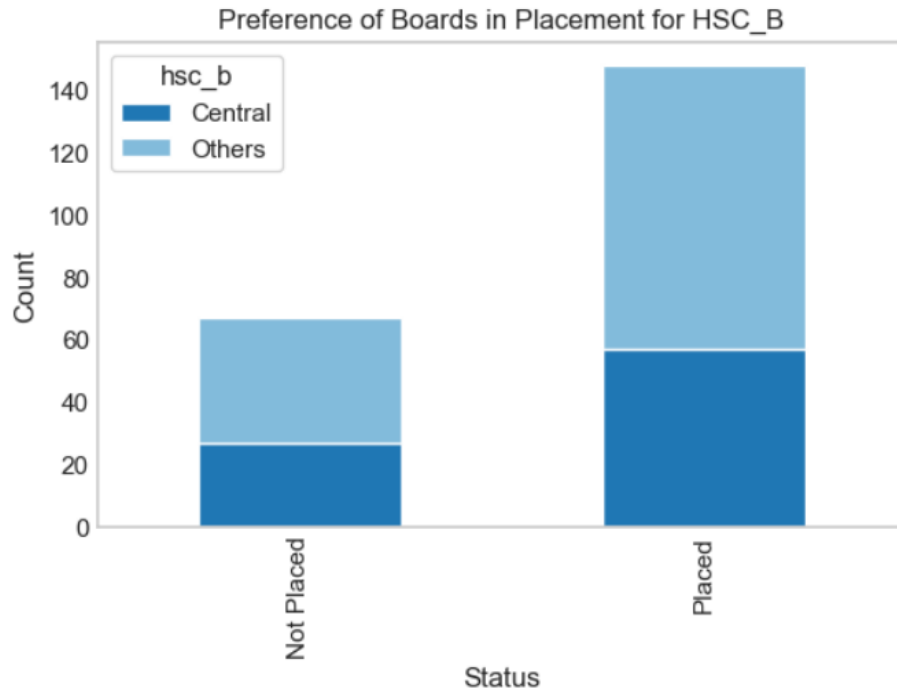
- 1) Students who did not get a placement were having a percentage less than 60% in their secondary and higher secondary education.
- 2) The preferred stream in placement process is commerce then science.



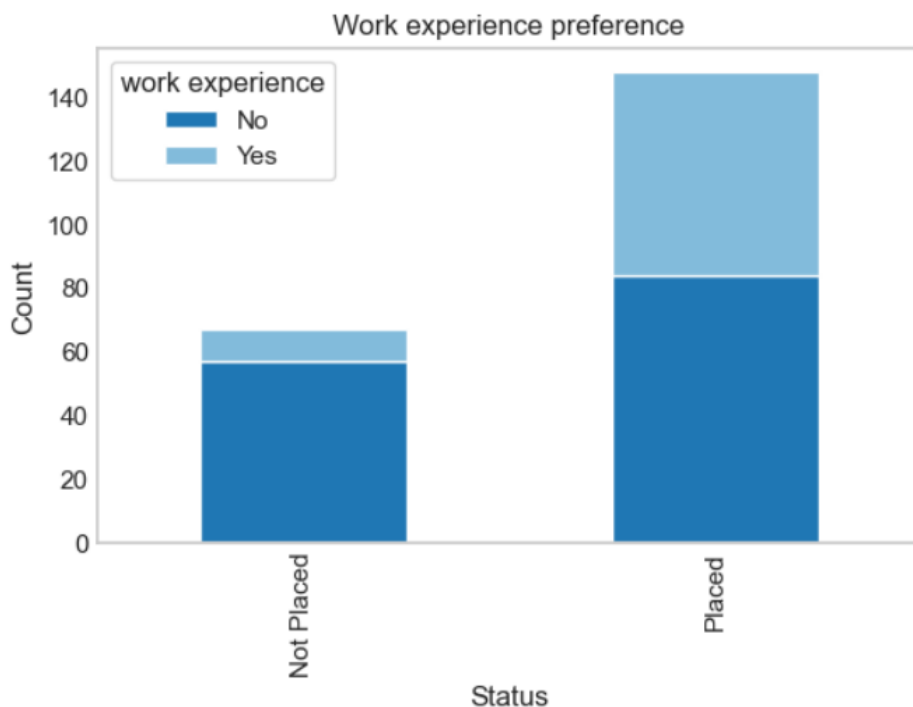
3) It seems like there is a slight gender gap between male and female students but it can be neglected as male students are higher in ratio.



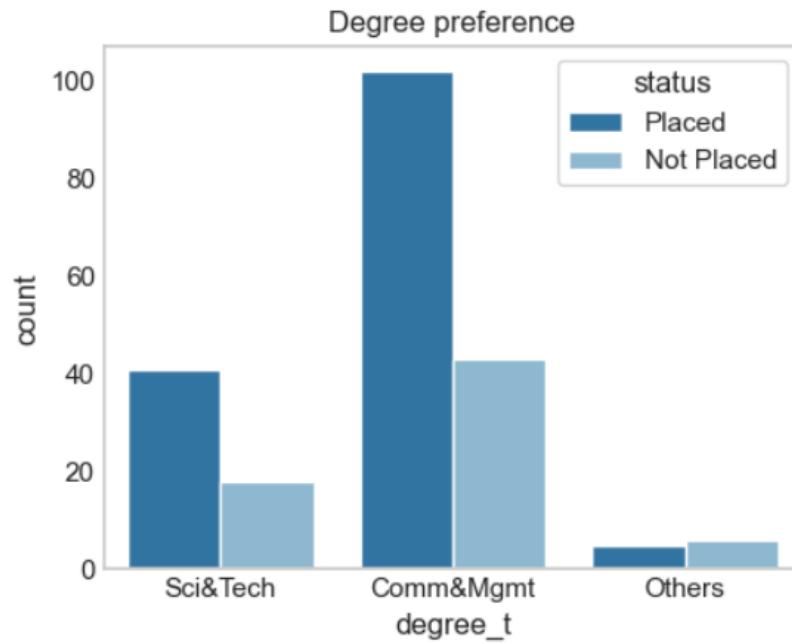
4) There is no such preference between secondary board as per the above plot.



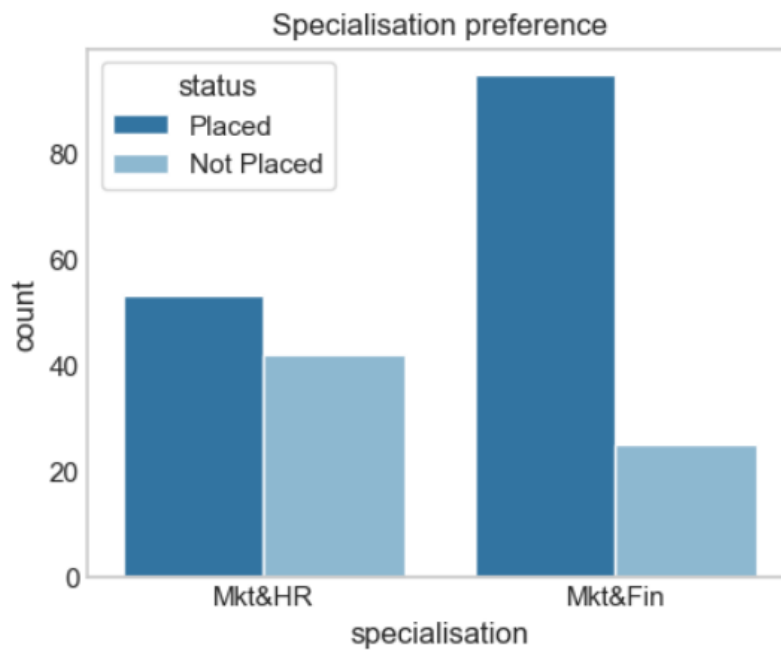
5) The above plot shows that companies are very likely to take students from non-central boards in the Higher Secondary Education of Students.



6) The above plot shows that most of students who didn't get a placement were not having work experience but also most the students who got placed were not having work experience. 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 platform.

```

IBM Watson Studio
Search in your workspaces

Projects / IBM_HC / College Placement Data Analysis

In [82]: model_uid = wml_client.repository.get_model_id(model_details)
         model_uid

Out[82]: '22b920ca-0cf4-4260-b01d-0f9ebe9b2311'

In [83]: #save meta
         deployment_props = {
             wml_client.deployments.ConfigurationMetaNames.NAME: DEPLOYMENT_NAME,
             wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
         }

In [84]: #Deploy
         deployment1 = wml_client.deployments.create(
             artifact_uid = model_uid,
             meta_props = deployment_props
         )

#####

Synchronous deployment creation for uid: '22b920ca-0cf4-4260-b01d-0f9ebe9b2311' started

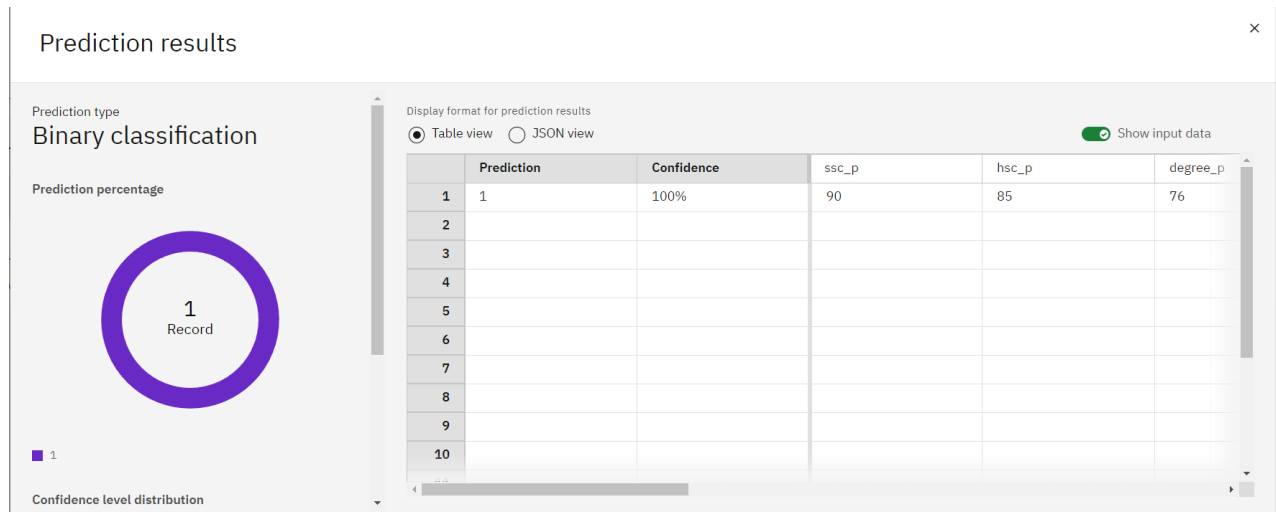
#####

initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready

-----
Successfully finished deployment creation, deployment_uid='5676a03e-bff1-42bf-82af-06853637c25b'
-----

```



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
1 from flask import Flask, render_template, request, jsonify
2 import requests
3 import pickle
4
5 app = Flask(__name__)
6
7 model_filename = 'classification_gnb.pkl'
8 with open(model_filename, 'rb') as file:
9     model = pickle.load(file)
10
11 prediction_statements = {
12     0: "Sorry, You cannot get a placement.",
13     1: "Congartulations! You can get a placement."
14 }
15
16 gender_mapping = {'F': 0, 'M': 1}
17 ssc_b_mapping = {'Central': 0, 'Others': 1}
18 hsc_b_mapping = {'Central': 0, 'Others': 1}
19 hsc_s_mapping = {'Arts': 0, 'Commerce': 1, 'Science': 2}
20 degree_t_mapping = {'Comm&Mgmt': 0, 'Others': 1, 'Sci&Tech': 2}
21 workex_mapping = {'No': 0, 'Yes': 1}
22 specialisation_mapping = {'Mkt&Fin': 0, 'Mkt&HR': 1}
23
24 @app.route('/')
25 def index():
26     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.

```
app.py x
app.py > predict
28 @app.route('/predict', methods=['POST'])
29 def predict():
30     ssc_p = float(request.form['ssc_p'])
31     hsc_p = float(request.form['hsc_p'])
32     degree_p = float(request.form['degree_p'])
33     etest_p = float(request.form['etest_p'])
34     mba_p = float(request.form['mba_p'])
35     gender = request.form['gender']
36     ssc_b = request.form['ssc_b']
37     hsc_b = request.form['hsc_b']
38     hsc_s = request.form['hsc_s']
39     degree_t = request.form['degree_t']
40     workex = request.form['workex']
41     specialisation = request.form['specialisation']
42
43     # Convert categorical data to encoded values using mapping dictionaries
44     gender_encoded = gender_mapping[gender]
45     ssc_b_encoded = ssc_b_mapping[ssc_b]
46     hsc_b_encoded = hsc_b_mapping[hsc_b]
47     hsc_s_encoded = hsc_s_mapping[hsc_s]
48     degree_t_encoded = degree_t_mapping[degree_t]
49     workex_encoded = workex_mapping[workex]
50     specialisation_encoded = specialisation_mapping[specialisation]
51
52     features = [ssc_p, hsc_p, degree_p, etest_p, mba_p, gender_encoded, ssc_b_encoded, hsc_b_encoded,
53                hsc_s_encoded, degree_t_encoded, workex_encoded, specialisation_encoded]
54
```

Finally imported all the requests and connected to the Watson API.

```
55 import requests
56
57 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
58 API_KEY = "k8UUMJsw5607F7TaevmyW74_AoCOetZHX0TUzUjX10"
59 token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":API_KEY, "grant_type":
60 mltoken = token_response.json()["access_token"]
61
62 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
63
64 payload_scoring = {"input_data": [{"fields": ["ssc_p","hsc_p","degree_p","etest_p","mba_p","gender","ssc_b","hsc_b","hsc_s","degree_t","workex","specialisation"],
65 "values": [[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]]}]
66
67 try:
68     response = requests.post('https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/5676a03e-bff1-42bf-82af-06853637',
69     print("Scoring response")
70     print(response.json())
71     if response.status_code == 200:
72         api_data = response.json()
73         if 'predictions' in api_data and len(api_data['predictions']) > 0:
74             prediction_data = api_data['predictions'][0]
75             if 'fields' in prediction_data and 'values' in prediction_data:
76                 prediction_values = prediction_data['values'][0]
77                 prediction = prediction_values[0]
78                 prediction_statement = prediction_statements.get(prediction, "Unknown")
79                 return render_template('input.html', prediction=prediction_statement)
80             else:
81                 return "Error: Missing 'fields' or 'values' in prediction data."
82         else:
83             return "Error: No predictions found in API response."
84     else:
85         return "Error: Unable to get prediction from IBM Cloud."
86 except requests.exceptions.RequestException as e:
87     return "Error: " + str(e)
88
89 if __name__ == '__main__':
90     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:-

The screenshot shows a web browser window with the URL `127.0.0.1:5000/predict`. The page has a dark blue background. On the left, a sidebar contains the text "Project By - Yash Kumar Shukla" with an email address `shukla02yash@gmail.com`. Below this, it says "About the project:-" and describes the application as a campus placement prediction web application using a Naive Bayes Algorithm with 93% accuracy. It also lists "Tools used to make the project:-" with logos for Flask, Django, Python, and Jupyter. The main content area is titled "Campus Placement Prediction" and contains a form with the following fields: "Secondary Edu. Percentage: 42", "Higer Sec. Edu. Percentage: 33", "Degree Percentage: 56", "Employability Test Percentage: 64", "MBA Percentage: 56", "Gender: Female", "SSC Board: Central", "HSC Board: Central", "HSC Stream: Arts", "Degree Type: Comm&Mgmt", "Work Experience: No", and "MBA Specialisation: Mkt&Fin". A green "PREDICT" button is below the form. The "Output:-" section shows the message "Sorry, You cannot get a placement."

The screenshot shows the same web browser window with the URL `127.0.0.1:5000/predict`. The sidebar content is identical to the first screenshot. The main content area, titled "Campus Placement Prediction", shows the form with updated values: "Secondary Edu. Percentage: 93", "Higer Sec. Edu. Percentage: 95", "Degree Percentage: 88", "Employability Test Percentage: 89", "MBA Percentage: 82", "Gender: Male", "SSC Board: Central", "HSC Board: Others", "HSC Stream: Arts", "Degree Type: Comm&Mgmt", "Work Experience: Yes", and "MBA Specialisation: Mkt&Fin". The green "PREDICT" button is still present. The "Output:-" section now shows the message "Congartulations! You can get a placement."