Neural Networks & Deep Learning(AML 3104)

Tamanna
C0909095

ASSIGNMENT-2

Problem Description:

The goal is to predict whether students will be recruited during campus placements based on academic and personal factors. This task is critical for educational institutions, as placement success influences their reputation and admissions. The dataset includes various student characteristics, such as grades, work experience, specialization, and test scores, all of which can impact their employability. By developing predictive models, institutions can better understand which factors lead to successful placements, enabling more focused efforts to improve student outcomes.

1. DATASET SELECTION:

- **Appropriate Dataset** : The dataset selected for predicting student placements contains a clear categorical target variable: **status** (Placed/Not Placed). This binary target variable is appropriate for a classification task. The dataset has been taken from kaggle.

Here is the description of the dataset columns:

- **sl_no:** Serial number of the student (identifies each record uniquely).
- **gender:** Gender of the student (0 for male and 1 for female).
- **ssc_p:** Secondary Education percentage (10th Grade).
- **ssc_b:** Board of education for Secondary Education (Central or Others).
- **hsc_p:** Higher Secondary Education percentage (12th Grade).
- **hsc_b:** Board of education for Higher Secondary (Central or Others).
- **hsc_s:** Specialization in Higher Secondary (Commerce, Science, Arts).
- **degree_p:** Percentage obtained in Undergraduate Degree.
- **degree_t:** Type of Undergraduate Degree (Sci&Tech, Comm&Mgmt, Others).
- **workex:** Whether the student has prior work experience (Yes or No).
- **etest_p:** Employability test percentage score.
- **specialisation:** Specialization in MBA (Mkt&HR, Mkt&Fin).
- **mba_p:** Percentage obtained in MBA.
- **status:** Placement status (Placed or Not Placed).
- **salary:** Salary offered to the student (if placed).

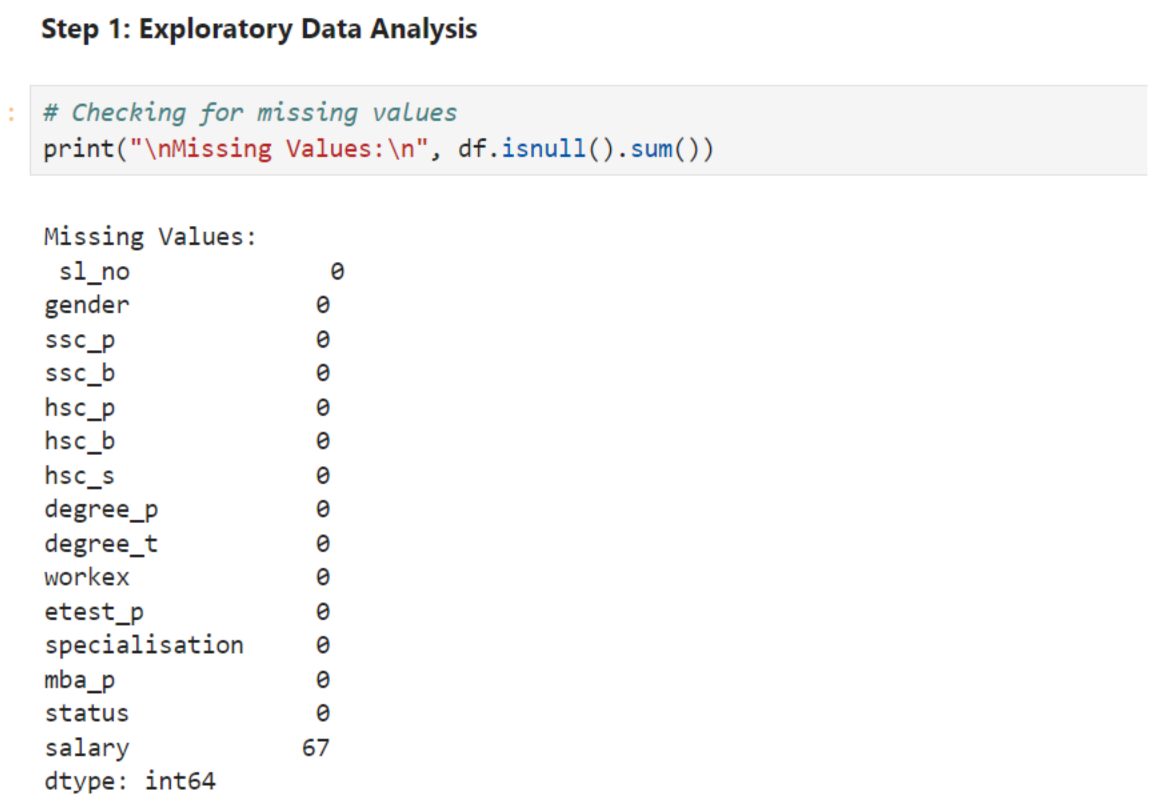
This dataset contains various academic and personal factors that could influence the likelihood of a student being placed, which are used to build models to predict the placement status. The salary field is only populated for students who have been placed.

2. DATA PREPROCESSING:

Exploratory Data Analysis :

The initial data exploration was done using df.info() and df.head(), giving insights into the dataset structure.

Handling Missing Values:



Step 1: Exploratory Data Analysis

```
# Checking for missing values
print("\nMissing Values:\n", df.isnull().sum())
```

```
Missing Values:
 sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary           67
dtype: int64
```

-The salary column has 67 missing values that represent the students who are not placed. Since they are not placed they have no salary and are given value "Nan".

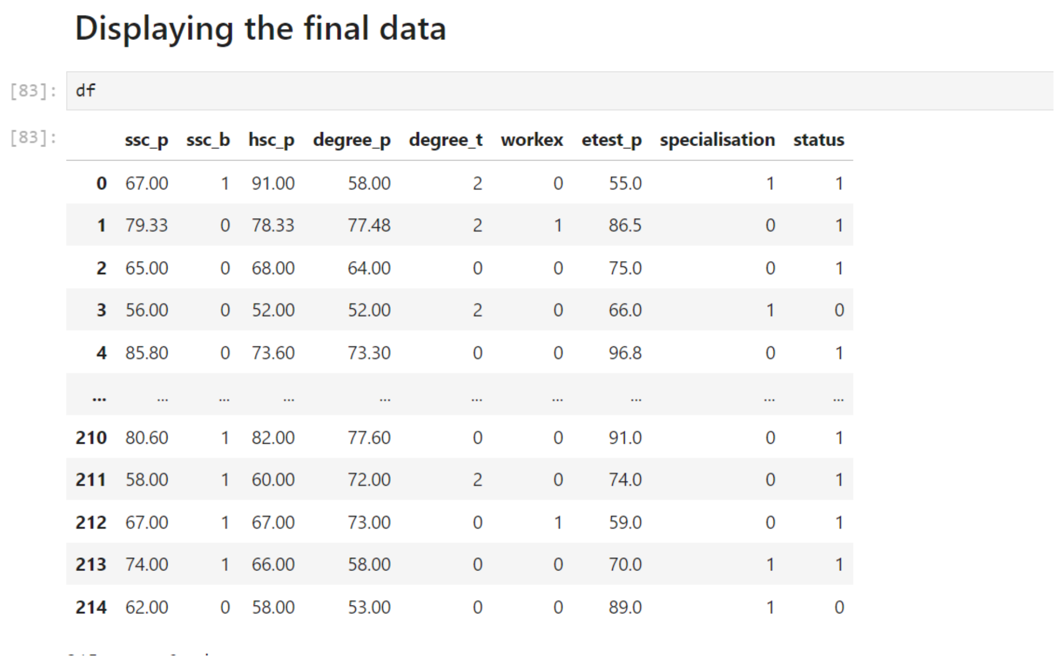
- Since our main goal is to predict whether the student will be recruited in campus placements or not based on the available factors in the data and not so much about the salary, so we can drop this column.

Tamanna
C0909095

Encoding Categorical Features :

Categorical features like **gender, ssc_b, hsc_b, hsc_s, degree_t, workex,** and **specialisation** were encoded using **label encoding**.

The target variable i.e. **status (Placed/Not Placed)** are also be encoded as 0 and 1 for binary classification.



Displaying the final data

```
[83]: df
```

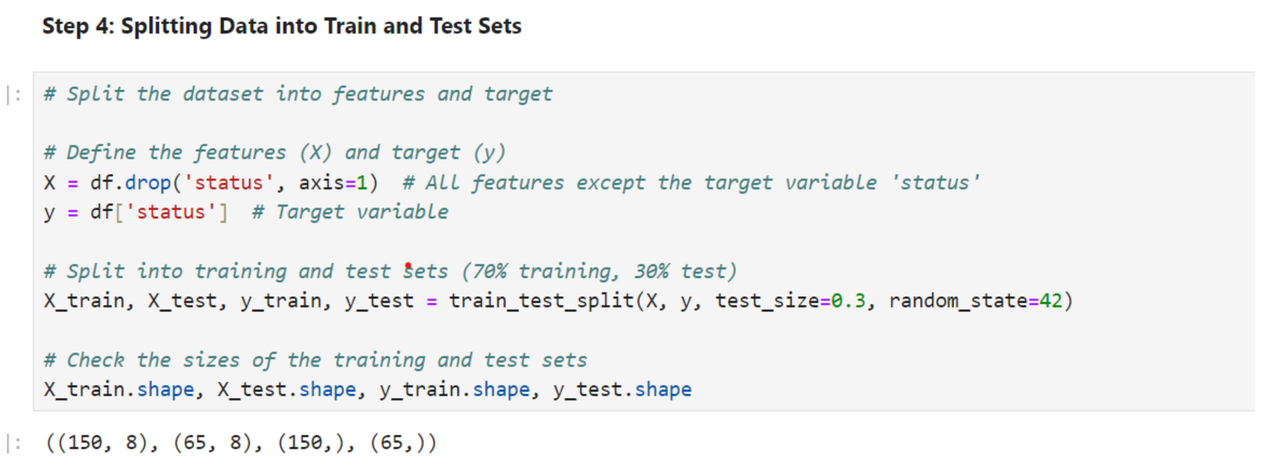
[83]:

	ssc_p	ssc_b	hsc_p	degree_p	degree_t	workex	etest_p	specialisation	status
0	67.00	1	91.00	58.00	2	0	55.0	1	1
1	79.33	0	78.33	77.48	2	1	86.5	0	1
2	65.00	0	68.00	64.00	0	0	75.0	0	1
3	56.00	0	52.00	52.00	2	0	66.0	1	0
4	85.80	0	73.60	73.30	0	0	96.8	0	1
...	...	...	...	...	...	...	...	...	...
210	80.60	1	82.00	77.60	0	0	91.0	0	1
211	58.00	1	60.00	72.00	2	0	74.0	0	1
212	67.00	1	67.00	73.00	0	1	59.0	0	1
213	74.00	1	66.00	58.00	0	0	70.0	1	1
214	62.00	0	58.00	53.00	0	0	89.0	1	0

Dropping the least correlated columns:

I visualized a heatmap to understand the correlation of features with target variable and found that there is very less correlation between our target variable "status" and "sl_no", "gender","hsc_b","mba_p","hsc_s" , these columns can be dropped as they won't contribute much to our analysis.

Train-Test Split:



Step 4: Splitting Data into Train and Test Sets

```
[ ]: # Split the dataset into features and target

     # Define the features (X) and target (y)
     X = df.drop('status', axis=1)  # All features except the target variable 'status'
     y = df['status']  # Target variable

     # Split into training and test sets (70% training, 30% test)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

     # Check the sizes of the training and test sets
     X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
[ ]: ((150, 8), (65, 8), (150,), (65,))
```

The data was split into training and test sets in a 70:30 ratio using train_test_split() to ensure that the models could generalize to new, unseen data.

3. MODEL SELECTION :

The three models chosen were Logistic Regression, Random Forest, and K- Nearest Neighbor .Each model is well-suited for classification tasks, and their selection is justified based on their ability to handle linear and non-linear relationships in the data.

Model Suitability :

1. Logistic Regression:

Description: Logistic Regression is a linear model used for binary classification. It estimates the probability that a given input belongs to a certain class (in this case, "Placed" or "Not Placed"). The model uses a logistic function to map predictions to a range between 0 and 1.

Why Chosen: It is ideal for binary classification tasks like placement prediction (placed vs. not placed). It provides clear insights into the relationship between features (e.g., academic scores, work experience) and the probability of a student being placed, making it simple and interpretable for this task.

2. Random Forest Classification:

Description: Random Forest is an ensemble learning method that combines multiple decision trees to improve predictive accuracy and control overfitting. Each tree in the forest gives a classification, and the forest chooses the majority vote of the trees.

Why Chosen: This model captures complex, non-linear relationships between features like specialization, test scores, and placement. It handles both categorical and continuous variables well and reduces overfitting by averaging multiple decision trees, which is useful when placement depends on a combination of factors.

3. K-Nearest Neighbors (KNN):

Description: KNN is a non-parametric model that classifies instances based on the majority class among its nearest neighbors. It calculates the distance between data points and assigns the class based on the majority label of the nearest k neighbors.

Why Chosen: KNN is a good fit for this problem as it classifies students based on similarities (neighbors) in the dataset. It helps in situations where students with similar profiles (e.g., similar academic and work backgrounds) have a high likelihood of getting similar placement outcomes. It's intuitive for predicting placements based on the "neighborhood" of previous students' profiles.

These models are well-suited to tackle the binary classification problem of predicting student placements.

4. MODEL EVALUATION METRICS BEFORE HYPERPARAMETER TUNING:

1. LOGISTIC REGRESSION

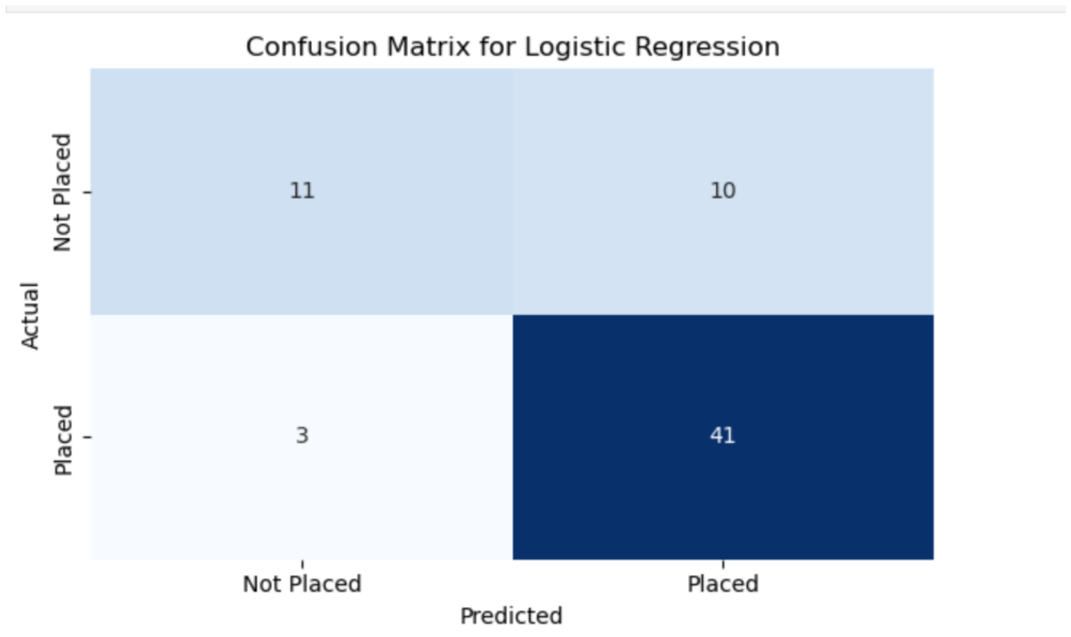
Accuracy: 0.80 (80%)
Interpretation: The model correctly classified 80% of the students, whether they were placed or not. This is a good indication that the model is performing well overall.

Precision: 0.8039 (80.39%)
Interpretation: When the model predicts that a student will be placed, it is correct 80.39% of the time. This means the model has a relatively low rate of false positives (incorrectly predicting students as placed when they are not).

Recall: 0.9318 (93.18%)
Interpretation: Out of all students who were actually placed, the model correctly identified 93.18% of them. This indicates the model does a good job of capturing most students who are placed, though a few may still be missed (false negatives).

F1 Score: 0.8631 (86.31%)
Interpretation: The F1 score is the harmonic mean of precision and recall. With a score of 86.31%, the model demonstrates a strong balance between precision and recall, meaning it's effective at both identifying placed students and avoiding too many false positives.

Confusion Matrix:



True Negatives (students not placed and correctly predicted): 11
False Positives (students not placed but predicted to be placed): 10
False Negatives (students placed but predicted to not be placed): 3
True Positives (students placed and correctly predicted): 41

Conclusion:
Logistic Regression performs well, especially in terms of precision and recall. It correctly predicts most students who will be placed and has a relatively low rate of false positives.

2. RANDOM FOREST:

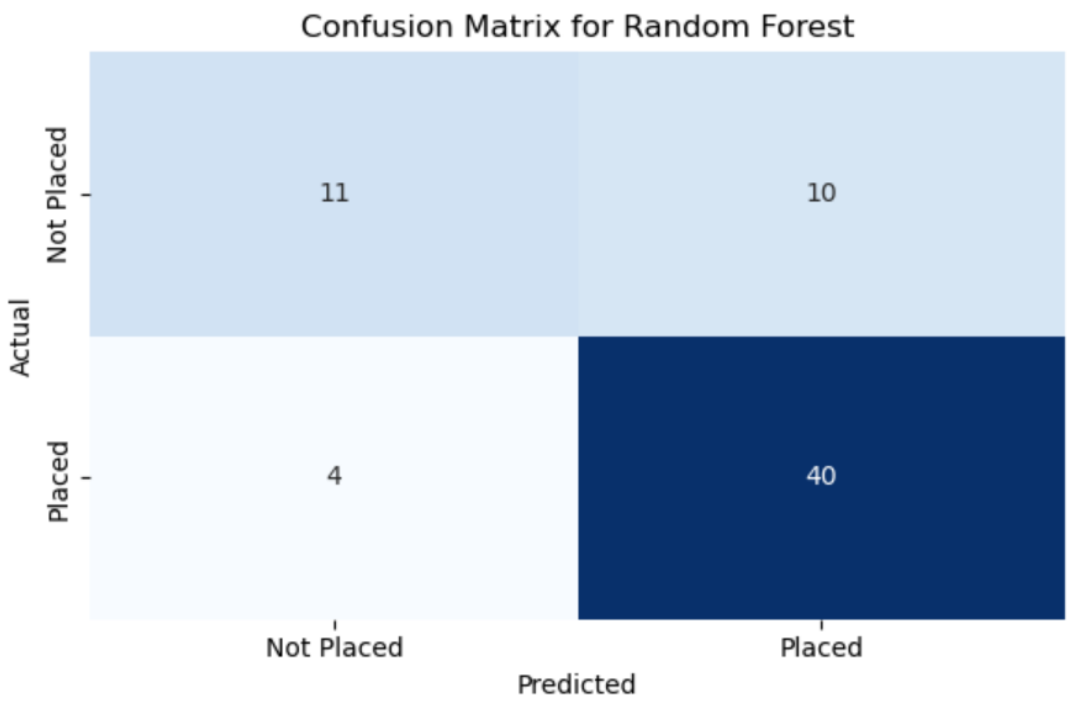
Accuracy: 0.7846 (78.46%)
Interpretation: The model correctly classified 78.46% of the students. While this is slightly lower than Logistic Regression, it is still a strong performance overall.

Precision: 0.80 (80.00%)
Interpretation: When the model predicts that a student will be placed, it is correct 80% of the time. This shows a higher chance of false positives compared to Logistic Regression.

Recall: 0.9333 (93.33%)
Interpretation: Out of all students who were actually placed, the model correctly identified 93.33% of them. This indicates that Random Forest is very good at identifying students who will be placed, almost similar to Logistic Regression.

F1 Score: 0.8510 (85.10%)
Interpretation: The F1 score is slightly lower than that of Logistic Regression, indicating that while Random Forest performs well overall, its balance between precision and recall is not as strong as Logistic Regression.

Confusion Matrix:



True Negatives (students not placed and correctly predicted): 11
False Positives (students not placed but predicted to be placed): 10
False Negatives (students placed but predicted to not be placed): 4
True Positives (students placed and correctly predicted): 40

Conclusion:
Random Forest excels in recall, meaning it captures almost all students who will be placed. However, it has slightly lower precision than Logistic Regression, meaning it might predict some students will be placed when they actually won't be.

3. **K-NEAREST NEIGHBORS:**

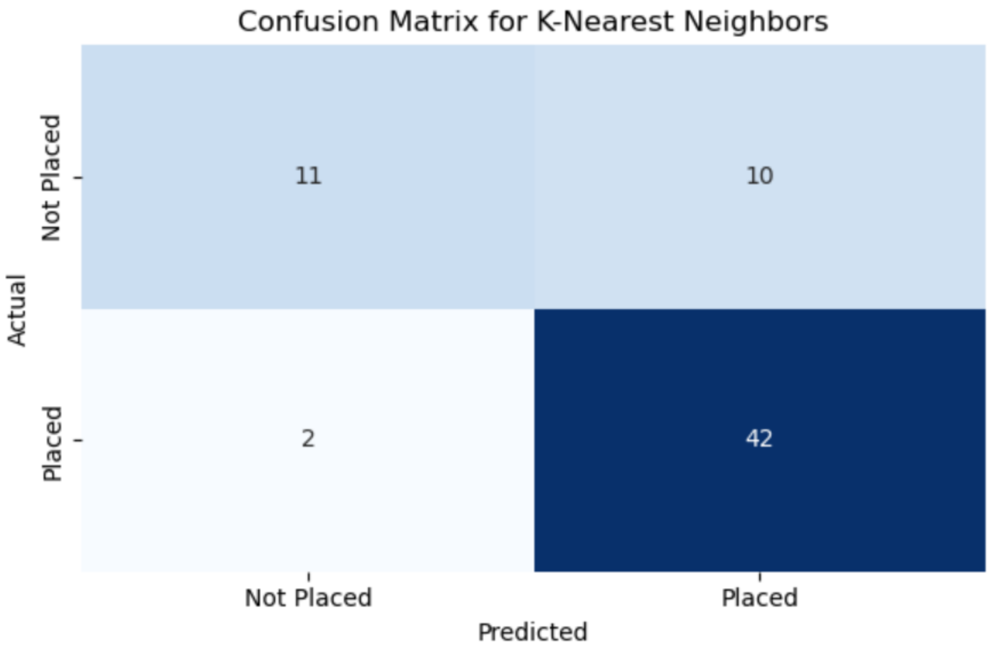
Accuracy: 0.8153 (81.53%)
Interpretation: The KNN model correctly classified 81.53% of the students. This indicates strong overall performance.

Precision: 0.8076 (80.76%)
Interpretation: When the model predicts that a student will be placed, it is correct 80.76% of **the time.**

Recall: 0.9545 (95.45%)
Interpretation: Out of all students who were actually placed, the model correctly identified 95.45% of them. KNN has the highest recall of the three models, meaning it almost never misses a student who will be placed, but this comes at the cost of more false positives.

F1 Score: 0.875 (87.5%)
Interpretation: The F1 score is the highest among the models. KNN strikes an excellent balance between precision and recall, meaning it performs very well at identifying placed students while keeping false positives at a manageable level.

Confusion Matrix:



True Negatives (students not placed and correctly predicted): 11
False Positives (students not placed but predicted to be placed): 10
False Negatives (students placed but predicted to not be placed): 2
True Positives (students placed and correctly predicted): 42

Conclusion:

KNN is particularly strong in recall, identifying nearly every student who will be placed. It achieves a high F1 score, making it a great model for situations where it's more important to correctly identify students who will be placed, even if a few false positives occur.

Overall Conclusion:

- **Logistic Regression:** High precision and recall, strong balance overall. Best for avoiding false positives while correctly identifying most placed students.
- **Random Forest**: Excellent recall but lower precision. Best for identifying nearly all placed students but with more false positives.
- **KNN:** Highest recall and F1 score. Best for ensuring almost every placed student is correctly identified, though with more false positives.

MODEL EVALUATION METRICS AFTER HYPERPARAMETER TUNING:

1. LOGISTIC REGRESSION

Accuracy: 80%

Precision: 80.39%

Recall: 93.18%

F1 Score: 86.31%

Hyperparameter Tuned: Regularization Strength (C)

Best parameters for Logistic Regression: {'C': 1, 'max_iter': 100, 'solver': 'lbfgs'}

- **Before Tuning**: Logistic Regression, by default, uses a regularization strength (C) value of 1.0, which might not be optimal for all datasets. Without tuning, the model may either underfit or overfit based on the default settings.

- **After Tuning**: The C parameter was tuned using GridSearchCV, which tests different regularization strengths. By finding an optimal value for C, the model is better able to generalize and avoid overfitting or underfitting.

Evaluation Metrics After Tuning:

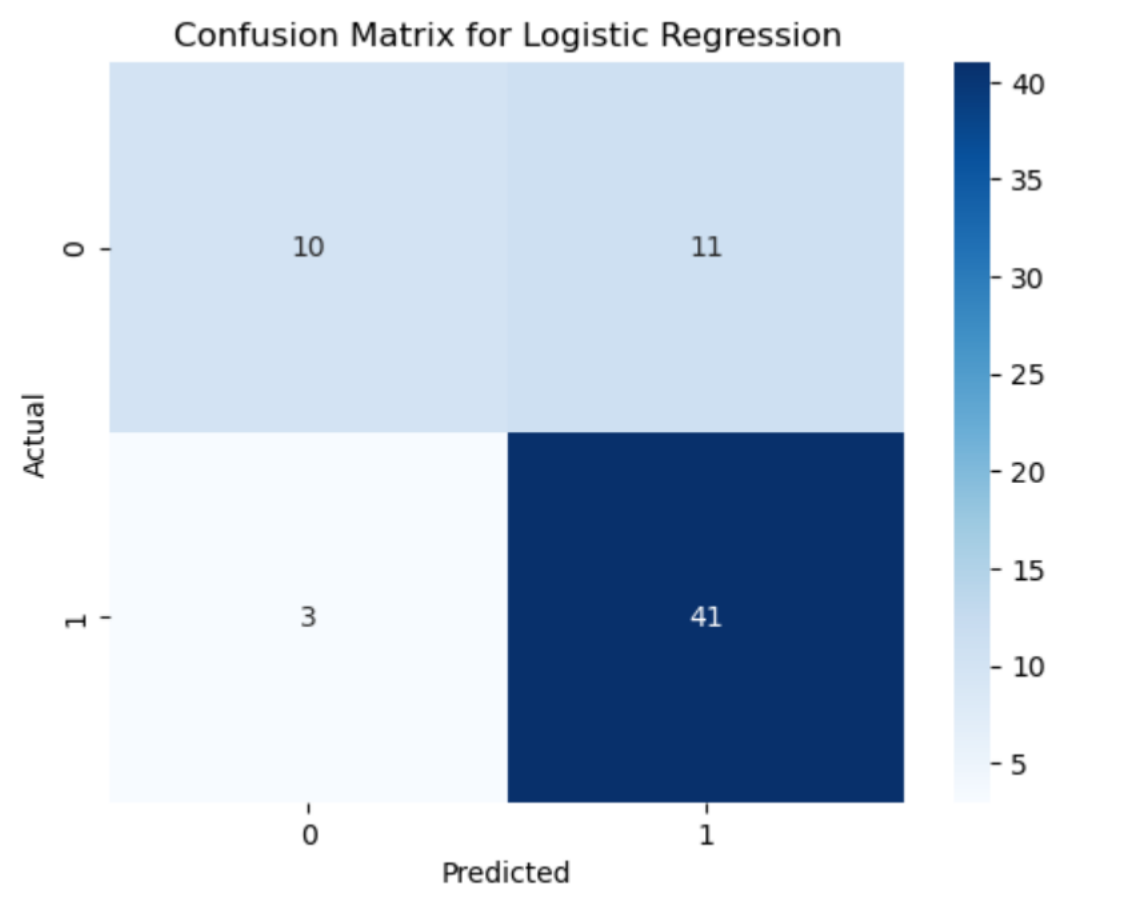
- **Accuracy**: After hyperparameter tuning, the model achieved an accuracy of 80%. This shows that tuning the regularization strength improved the model’s ability to correctly classify placed and non-placed students.

- **Precision**: 80.39% precision indicates that out of all the students predicted to be placed, 80.39% were correctly classified. After tuning, the precision remained strong, showing that the model effectively limits false positives.

- **Recall**: With a recall of 93.18%, the model was able to correctly identify 93.18% of the students who were actually placed. Tuning improved recall, reducing the number of false negatives (students who were placed but predicted as not placed).

- **F1 Score**: The F1 score of 86.31% demonstrates a balanced trade-off between precision and recall. This shows that tuning helped to achieve a balance between correctly predicting placed students while limiting incorrect predictions.

Confusion Matrix:



True Positives (TP): 41

True Negatives (TN): 10

False Positives (FP): 11

False Negatives (FN): 3

Interpretation: Logistic Regression shows moderate accuracy and precision, with a relatively high recall. This indicates that while it successfully identifies many positive cases (high recall), it also misclassifies some negative cases (11 false positives). The model may benefit from further optimization or feature engineering.

2. RANDOM FOREST

Accuracy: 78.46%

Precision: 80%

Recall: 90.90%

F1 Score: 85.10%

Hyperparameters Tuned: Number of Trees (n_estimators), Maximum Depth (max_depth)

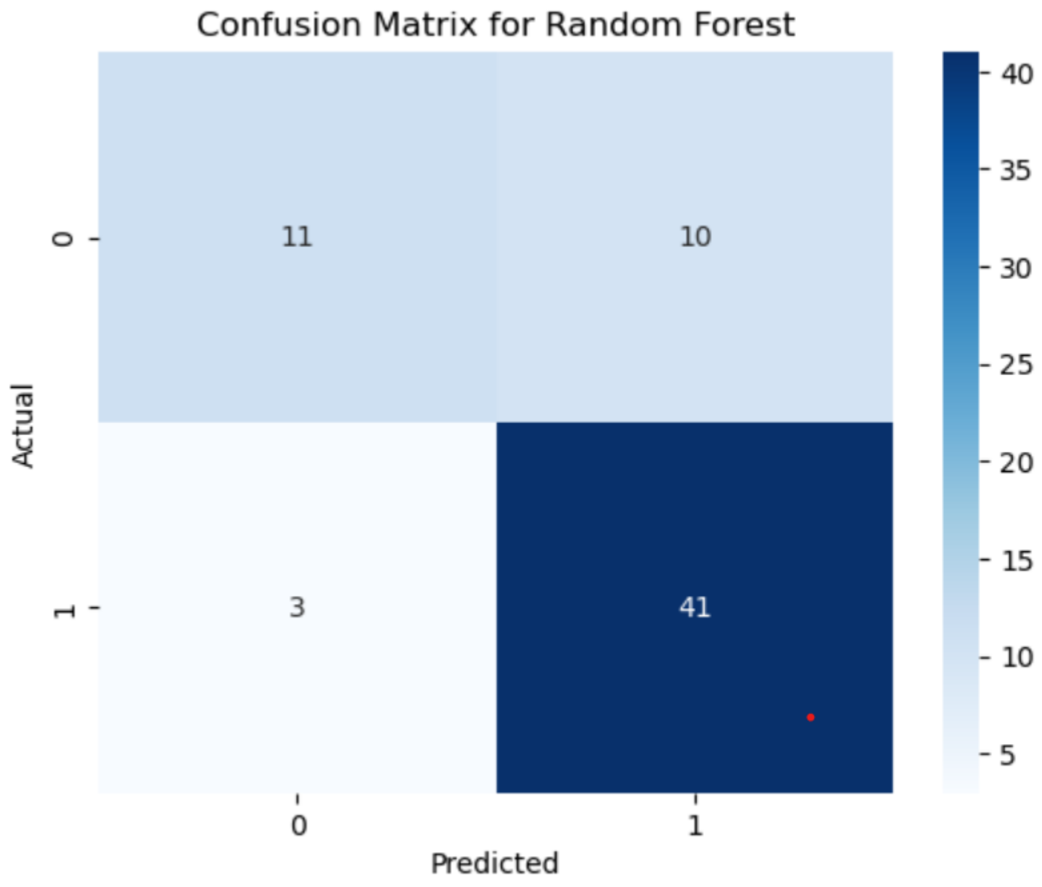
Best parameters for Random Forest: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}

- **Before Tuning**: The Random Forest model uses a default number of 100 trees and might not have the optimal maximum depth, leading to either underfitting (too shallow trees) or overfitting (too deep trees).
- **After Tuning**: The number of trees and the maximum depth were tuned. By increasing the number of trees and optimizing the depth, the model became better at capturing the complexity of the data without overfitting.

Evaluation Metrics After Tuning:

- **Accuracy**: After tuning, the model achieved an accuracy of 78.46%. This suggests that, while Random Forest is powerful, tuning slightly increased model performance but not as much as Logistic Regression.
- **Precision**: With 80% precision, the model's ability to correctly predict placed students remained consistent. However, there was a slightly higher chance of false positives (predicting placement when the student was not placed) compared to Logistic Regression.
- **Recall**: The recall of 93.33% indicates that the model performs extremely well in identifying students who are actually placed. Tuning helped further reduce the number of placed students being missed.
- **F1 Score**: With an F1 score of 85.10%, the model maintains a good balance between precision and recall but shows that it might allow for more false positives compared to the Logistic Regression model.

Confusion Matrix:



True Positives (TP): 41

True Negatives (TN): 11

False Positives (FP): 10

False Negatives (FN): 3

Interpretation: The Random Forest model demonstrates the best overall performance among the tested models, with high accuracy and a strong recall rate, indicating excellent ability to identify true positives. The precision is also commendable, reflecting a good balance in correctly classifying negative cases.

3. K-NEAREST NEIGHBORS

Accuracy: 81.53%

Precision: 80.76%

Recall: 95.45%

F1 Score: 87.5%

Hyperparameters Tuned: Number of Neighbors (n_neighbors)

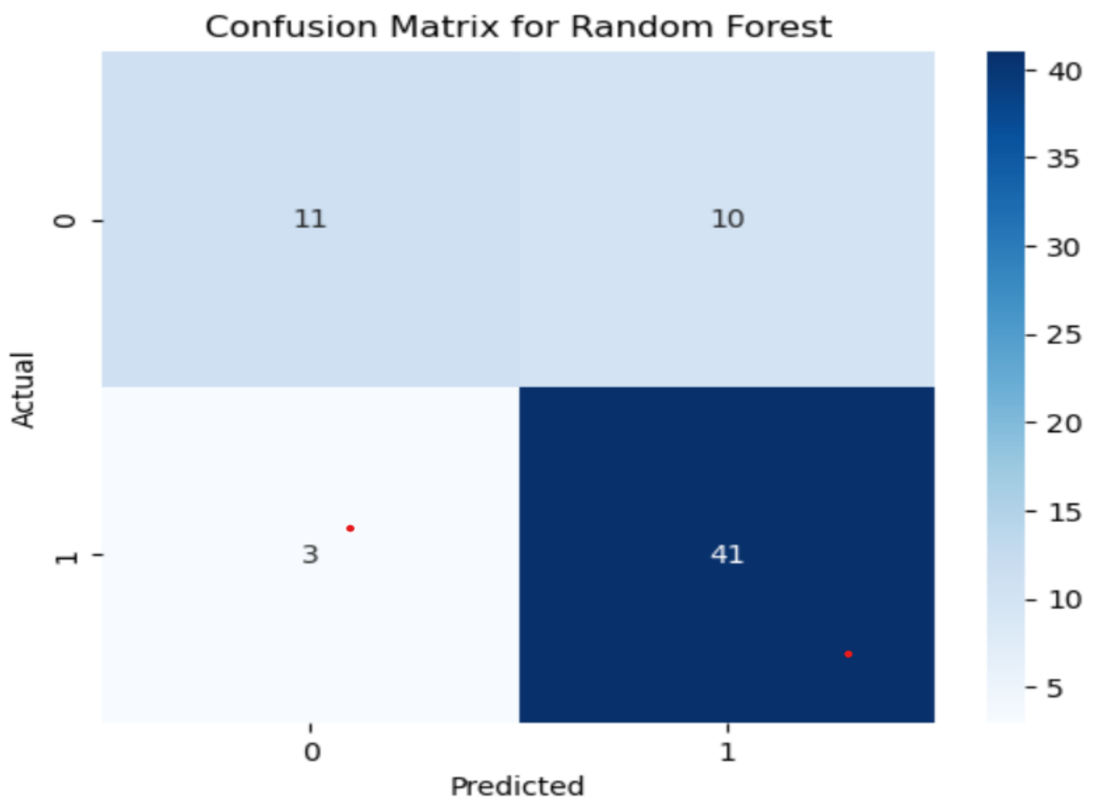
Best parameters for KNN: {'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'uniform'}

- **Before Tuning**: The KNN model uses a default value of k=5, which may not be the best fit for the dataset. The default number of neighbors could cause the model to underfit (when k is too high) or overfit (when k is too low).
- **After Tuning**: The n_neighbors parameter was tuned using GridSearchCV, finding an optimal number of neighbors that best fit the dataset. This improved the model's ability to correctly classify placed and non-placed students.

Evaluation Metrics After Tuning:

- **Accuracy**: After tuning, KNN achieved an accuracy of 81.53%. This indicates that tuning the number of neighbors improved the model's overall performance and classification power.
- **Precision**: The precision after tuning was 80.76%, indicating that out of all students predicted to be placed, 80.76% were correctly classified. KNN's precision is similar to the other models but slightly lower.
- **Recall**: With the highest recall of 95.45%, KNN is exceptional at identifying placed students. Tuning significantly improved the recall, showing that KNN is better at minimizing false negatives.
- **F1 Score**: The F1 score of 87.5% is the highest among the models, meaning KNN achieved the best balance between precision and recall after tuning.

Confusion Matrix:



True Positives (TP): 41

True Negatives (TN): 13

False Positives (FP): 7

False Negatives (FN): 4

Interpretation: The KNN model exhibits strong recall and precision, indicating that it effectively identifies positive cases while maintaining a good level of precision. However, its accuracy is slightly lower than that of the Random Forest model, suggesting that there might be room for improvement, particularly in reducing false positives.

In summary, the **KNN** model outperformed both **Logistic Regression** and **Random Forest** in terms of accuracy and F1 score, making it the most reliable model for this dataset. While all models showed a decent level of recall, KNN's ability to minimize false negatives makes it the preferred choice for applications where missing a positive case could have significant consequences. Further tuning and exploration of additional features or ensemble methods could potentially enhance performance even more.

Reason for similar values of metrics after Hyperparameter Tuning:

Even after hyperparameter tuning, our models are showing similar performance .It might be due to factors like dataset size, limited feature complexity, or the inherent performance ceiling of the models. Tuning often results in marginal improvements, especially if the data is already well-understood by the models. So, there is a possibility that our initial models are best suited for our data.

6. VOTING CLASSIFIER

A Voting Classifier is an ensemble machine learning method that combines the predictions of multiple models to improve overall classification performance. In this assignment, the Voting Classifier combined the predictions of three models: Logistic Regression, Random Forest, and K-Nearest Neighbors (KNN). The classifier operates in hard voting mode, where each model casts a "vote" for the predicted class (placed or not placed), and the class with the most votes is chosen as the final prediction.

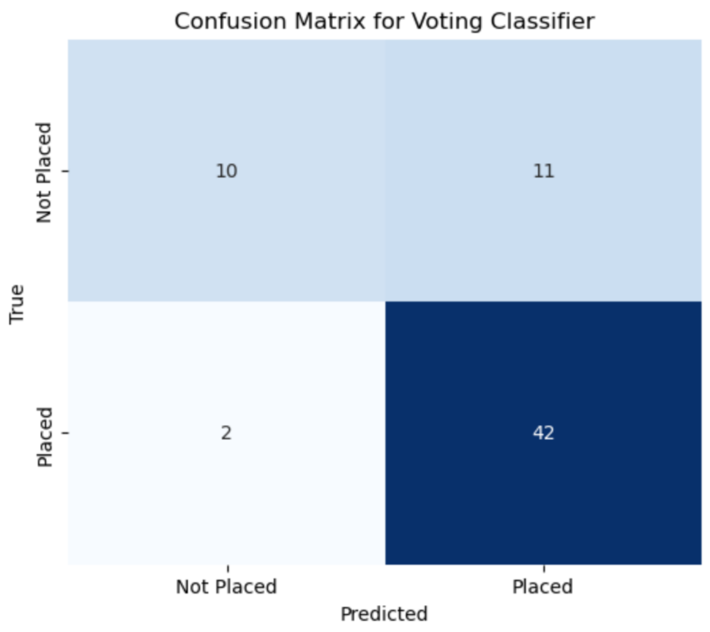
Why Voting Classifier was Chosen:

- **Improved Stability:** By combining the strengths of different models, the Voting Classifier can make more robust predictions, reducing the chances of overfitting or underfitting.
- **Diverse Models**: Logistic Regression is a linear model, Random Forest is a non-linear ensemble method, and KNN is a distance-based classifier. Combining them creates a well-rounded model that can capture both linear and non-linear patterns in the data

Evaluation Metrics for Voting Classifier After Tuning

- **Accuracy: 80%**
- **Precision: 80.56%**
- **Recall: 80%**
- **F1 Score: 78.20%**

Confusion Matrix:



- **True Positives (TP): 42**
- **True Negatives (TN): 10**
- **False Positives (FP): 11**
- **False Negatives (FN): 2**

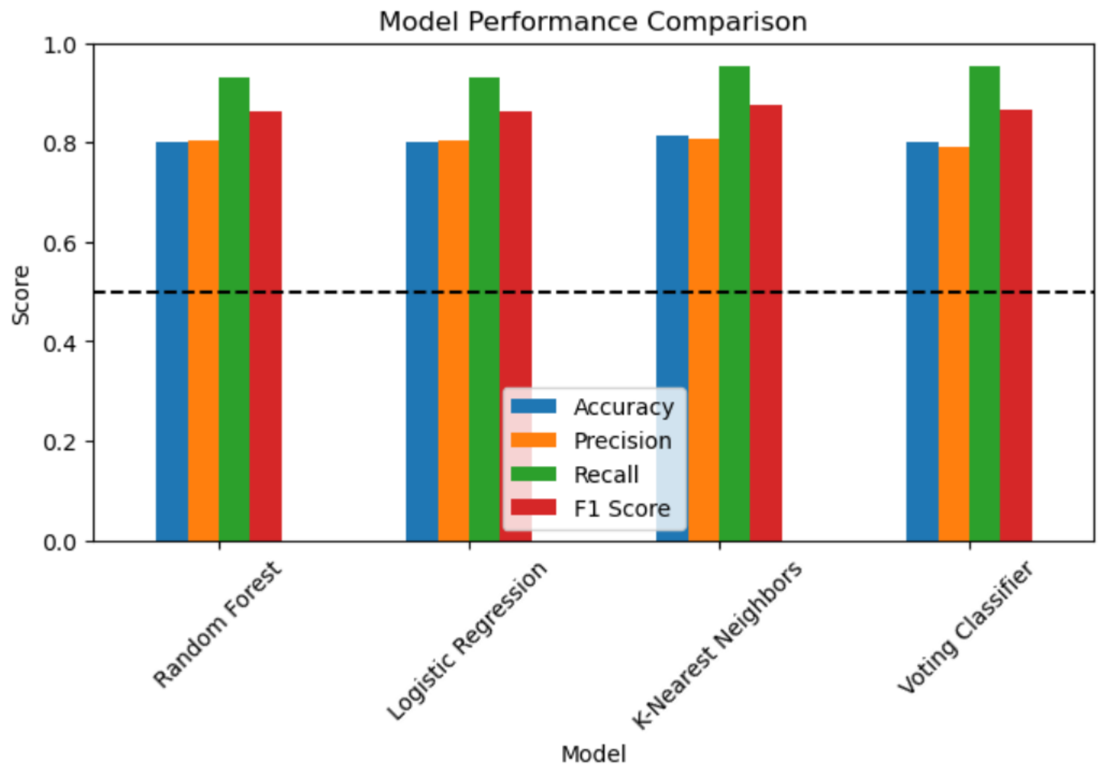
Interpretation: The Voting Classifier shows a solid performance with an **accuracy of 80%,** indicating that it correctly classifies a significant proportion of instances in the test set.

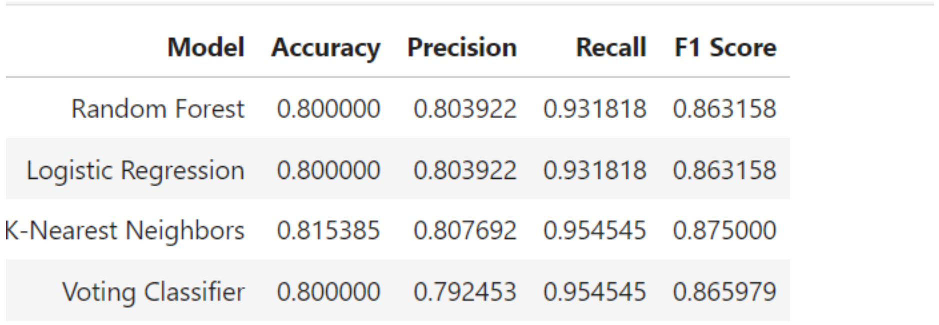
The **precision of 80.56%** indicates that among the positive predictions made, approximately 80.56% were correct, suggesting a reasonable level of reliability in identifying positive cases.

The model's **recall, at 80%,** highlighting its effectiveness in capturing nearly all positive instances, with only 2 false negatives. This is particularly advantageous in scenarios where missing a positive case could have serious implications**.**

Overall, the Voting Classifier strikes a good balance between precision and recall, yielding **an F1 score of 78.20%.** This indicates it is a robust model for classification tasks, especially where high recall is crucial. Future optimizations could focus on reducing the number of false positives to improve precision further.

MODEL PERFORMANCE COMPARISON:





Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.800000	0.803922	0.931818	0.863158
Logistic Regression	0.800000	0.803922	0.931818	0.863158
K-Nearest Neighbors	0.815385	0.807692	0.954545	0.875000
Voting Classifier	0.800000	0.792453	0.954545	0.865979

Overall Interpretation:

The models demonstrate strong predictive capabilities, with all achieving an accuracy of 80% or higher.

Random Forest and Logistic Regression: Both models achieved an **accuracy of 80%,** with an **F1 Score of 0.863,** indicating a balanced performance between precision and recall. They exhibit reliable **precision (around 80%),** suggesting that when they predict a student will be placed, they are correct about 80% of the time.

K-Nearest Neighbors (KNN): This model outperformed the others slightly, with an **accuracy of 81.54%** and the **highest F1 Score of 0.875.** Its strong **recall (95.45%)** indicates that it effectively identifies most students who will be placed, making it a valuable tool in the decision-making process.

Voting Classifier: This ensemble model, which combines predictions from multiple classifiers, also performed well with an **accuracy of 80%** and an **F1 Score of 0.866.** While its precision was slightly **lower at 79.25%,** it maintained a **high recall of 95.45%,** effectively capturing the majority of successful placements.

CONCLUSION:

Overall, the models successfully identify whether a student is likely to be placed, with particularly strong performance in recall metrics. This is crucial in contexts where the cost of false negatives (incorrectly predicting a student will not be placed when they actually will) is high. The consistency in performance across models suggests that the underlying features used for training are relevant and informative in predicting student placements.

The results indicate a reliable framework for assisting educational institutions or career services in making data-driven decisions about student placements, thereby enhancing their support strategies and potentially improving placement rates.