IE 6318 Data Mining and Analytics Data Mining Project
Sai Kiran Ganji (ID:1002081824)
Client Behavior Prediction on Term Deposit
Industrial Engineering
The University of Texas at Arlington

Abstract:

The main goal of this application-based project is to construct a prediction model to identify existing clients that have a higher chance to subscribe for a term deposit and focus marketing efforts on such clients. In this project, I will analyse the Bank lead's dataset and create various classification algorithms with full-end feature engineering and EDA. I will then find the best classification algorithm amongst the Naïve Bayes, Decision Trees, Random Forest, Nearest Neighbours, and Support Vector Machine for prediction.

Introduction:

One of a bank's main revenue streams is from term deposits. An investment in cash maintained at a financial institution is called a term deposit. Your funds are invested for a predetermined period of time, or term, at a predetermined rate of interest. The bank uses a variety of outreach strategies, including email marketing, digital marketing, telemarketing, and commercials, to pitch term deposits to its clientele. Campaigns for telephone marketing are still among the best ways to connect with consumers. But when these efforts are really carried out by hiring sizable call centers, they come with a hefty price tag. Therefore, in order to specifically target them with a call, it is imperative to identify the consumers who are most likely to convert beforehand. The data relates to a Portuguese financial institution's phone-based direct marketing activities. Predicting whether a consumer will sign up for a term deposit is the classification target (variable y).

Background:

The dataset is openly accessible for scholarly purposes. It has been picked up from the UCI Machine Learning with random sampling and a few additional columns. The full dataset was described and analysed in:

- S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology.
- In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.

Data Description:

The source of data is the Kaggle website. We have 4118 instances and 21 features. The information says there are no null values. Following are the attributes for the data :

1. Age : Age of the lead (numeric)
2. Job : type of job (Categorical)
3. Marital : Marital status (Categorical)
4. Education : Educational Qualification of the lead (Categorical)
5. Default: Does the lead has any default(unpaid)credit (Categorical)
6. Housing: Does the lead has any housing loan? (Categorical)
7. loan: Does the lead has any personal loan? (Categorical)
8. Contact: Contact communication type (Categorical)
9. Month: last contact month of year (Categorical)
10. day_of_week: last contact day of the week (categorical)
11. duration: last contact duration, in seconds (numeric).
12. campaign: number of contacts performed during this campaign and for this client (numeric)
13. pdays: number of days that passed by after the client was last contacted from a previous
14. campaign(numeric; 999 means the client was not previously contacted))
15. previous: number of contacts performed before this campaign and for this client (numeric)
16. poutcome: outcome of the previous marketing campaign (categorical)

The target variable is y which states if the client has subscribed to a term deposit.

Data Mining Experiments

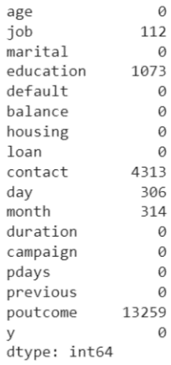
1. Data Pre-processing

A. Checking Duplicate Values

The presence of duplicate tuples is checked using ‘df.duplicated().sum()’ which returns the count of duplicate rows in the Data Frame df. If the result is greater than 0, it indicates that there are duplicate tuples present in the Data Frame; otherwise, it means there are no duplicates. We got 0 hence there are no duplicate tuples present.

B. Handling Missing Values

In the dataset null values are denoted by ‘unknown’ so we need to find and replace those to None which is done using replace method. After which we found null values using isna() method.



```
age              0
job            112
marital          0
education     1073
default          0
balance          0
housing          0
loan             0
contact       4313
day            306
month          314
duration         0
campaign         0
pdays            0
previous         0
poutcome     13259
y                0
dtype: int64
```

1. Null Values

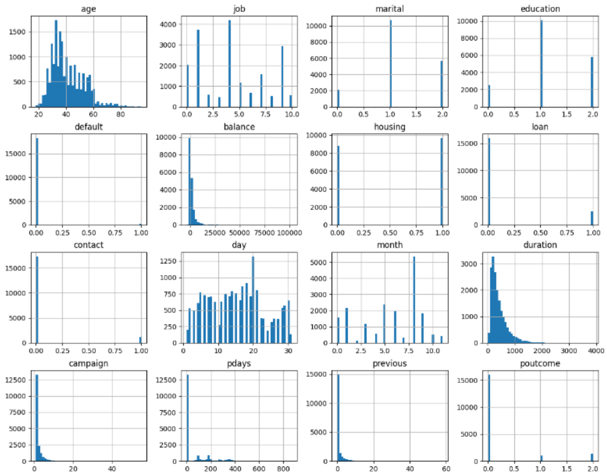
Based on the distribution of the available data, statistical techniques like mean, median, or mode imputation can be used to fill in the missing values. As the null values are present in categorical features I have replaced them using mode value of that feature.

C. Data conversion from categorical to numerical

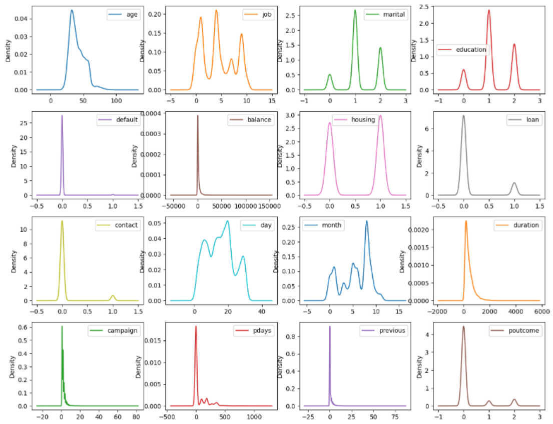
One typical stage in the data preparation pipeline is to convert categorical data to numerical representation. Numerous statistical models and machine learning methods demand numerical input, which makes this translation vital. The method used to transform categorical data into numerical representations can vary, and it is frequently determined by the characteristics of the category variables. Label Encoding is one such method which used here. It assigns a unique numerical label to each category.

D. Statistical Analysis

From the above figures we can see the distributions for each feature. Features namely 'age', 'default','balance','duration','campaign','pdays' and 'previous‘ are right-skewed. Features namely 'job', 'marital', 'education', 'housing', 'loan', 'contact', 'day', 'month' and 'poutcome' are in wave form, as those are having categorical values.



2. *Frequency Distribution Histogram*



3. *Frequency Distribution Histogram*

E. Feature Scaling

For right skewed data features standardization is done whereas for other features normalization using min-max scaler is used. Normalizing the numerical characteristics in the dataset and making sure they are on a consistent scale are the goals of feature scaling. This is especially crucial for machine learning methods that depend on the size of the input features, including gradient descent and distance-based techniques. With regard to the numerical features, we used the following feature scaling techniques:

- Z-score normalization, or standardization:

1. Method: Divide the result by the standard deviation after subtracting the mean.
2. Justification: Fits well with characteristics with a normal distribution.
3. Formula: $z=\sigma/x-\mu$

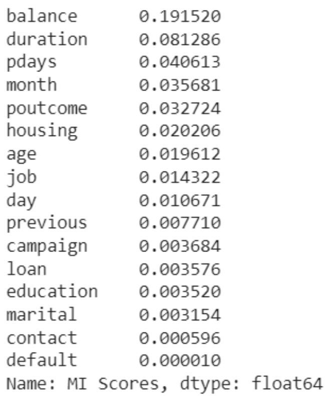
- Normalization using Min-Max Scaling:

1. Method: Set the features' scale to a predetermined range, usually [0, 1].
2. Justification: Beneficial in situations when characteristics deviate from a normal distribution and algorithms are sensitive to different magnitudes.

3. Formula: $$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

F. Feature Selection

Finding and keeping the most useful characteristics from the dataset is the aim of feature selection, which also aims to maximize computing efficiency, interpretability, and model performance. When working with high-dimensional data or datasets that contain a large number of possibly duplicate or unnecessary characteristics, this approach is very pertinent. In decision trees or ensemble techniques such as Random Forests, information gain is frequently linked to feature selection. When working with target variables that are categorical, it is very helpful. Finding the attributes that have the biggest impact on a decision tree model's predicted accuracy may be done efficiently via information gain. It assesses a feature's capacity to lessen ambiguity surrounding the target variable. Information gain via a decision tree-based model may be more suited if the dataset includes both numerical and category variables. Hence we have used mutual information gain method for feature selection with threshold of 0.01 by hit and trial method. After this step features Previous, campaign, loan, education, marital, contact and default are dropped to avoid overfitting.



```
balance      0.191520
duration     0.081286
pdays        0.040613
month        0.035681
poutcome     0.032724
housing      0.020206
age          0.019612
job          0.014322
day          0.010671
previous     0.007710
campaign     0.003684
loan         0.003576
education    0.003520
marital      0.003154
contact      0.000596
default      0.000010
Name: MI Scores, dtype: float64
```

4. *MI Scores for each feature*

1. Data Mining Models

For machine learning, it is standard practice to divide a dataset into training and testing sets using the train_test_split function in scikit-learn. With the help of this function, which divides the data into two subsets at random, you may train your model on one subset and assess its performance on the other.

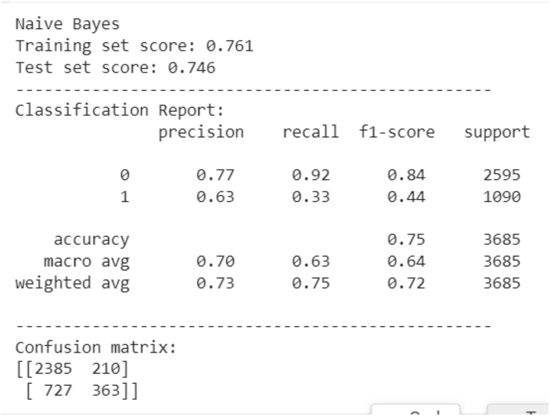
Here's a breakdown of the parameters:

- X: The feature matrix.
- y: The target variable (labels or values you are trying to predict).
- test_size: The proportion of the dataset to include in the test split. It can be a float (representing the proportion) or an integer (representing the absolute number of test samples). Common values are 0.2 for a 20% test set and 0.3 for a 30% test set.
- random_state: If provided, it ensures reproducibility by fixing the random seed. This means that running the code multiple times with the same random_state will result in the same split.

x and y are the feature matrix and target variable, respectively. train_test_split is used to split the dataset into training and testing sets. I gave train_size=0.8 specifies that 80% of the data should be used for training, and 20% for testing. random_state=42 ensures reproducibility by fixing the random seed. The same seed will produce the same split each time the code is run. shuffle=True indicates that the data should be randomly shuffled before splitting. This is important to ensure that the training and testing sets are representative of the overall distribution of the data.

A. Naive Bayes

The probabilistic classification method Naive Bayes is based on the Bayes theorem. In fact, it frequently works effectively despite its simplicity, particularly in text categorization and spam filtering. Although it makes calculation easier, the "naive" assumption of Naive Bayes is that features are conditionally independent given the class label. However, this assumption does not always apply in practical situations. A Gaussian Naive Bayes model has had its hyperparameters tuned, and I have assessed the model's performance using both training and test sets. The best-tuned model is then utilized for evaluation, with hyperparameter tweaking dependent on the var_smoothing parameter.

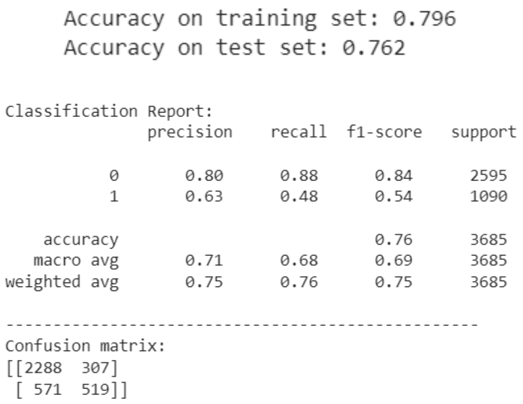


```
Naive Bayes
Training set score: 0.761
Test set score: 0.746
-------------------------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.92      0.84      2595
           1       0.63      0.33      0.44      1090

    accuracy                           0.75      3685
   macro avg       0.70      0.63      0.64      3685
weighted avg       0.73      0.75      0.72      3685

-------------------------------------------------
Confusion matrix:
[[2385  210]
 [ 727  363]]
```

4. *Naive Bayes Results*

B. Decision Tree

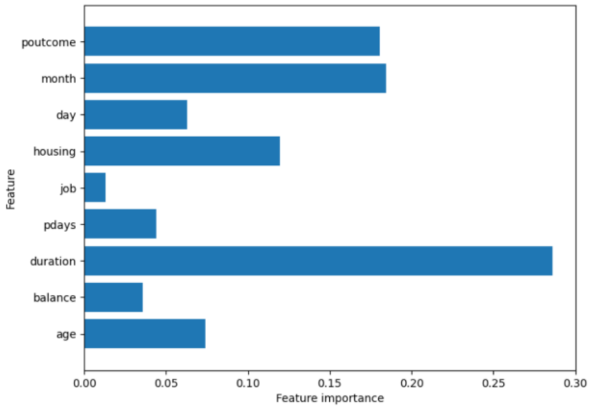
A supervised machine learning approach used for both regression and classification problems is called a decision tree. Recursively dividing the dataset into subgroups according to the most important attribute at each stage is how it operates. Through this process, a structure resembling a tree is created, with each internal node denoting a choice made in response to a feature, each branch representing a decision's result, and each leaf node representing the final prediction. Decision trees identify the optimal feature and threshold for separating the data at each node using a splitting criterion (e.g., mean squared error for regression or Gini impurity for classification). Until a halting condition is satisfied, the algorithm iteratively divides the data according to the chosen characteristics and thresholds (e.g., maximum depth attained, minimum samples per leaf, or no more improvement in impurity).



```
      Accuracy on training set: 0.796
      Accuracy on test set: 0.762


Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.88      0.84      2595
           1       0.63      0.48      0.54      1090

    accuracy                           0.76      3685
   macro avg       0.71      0.68      0.69      3685
weighted avg       0.75      0.76      0.75      3685

--------------------------------------------------
Confusion matrix:
[[2288  307]
 [ 571  519]]
```

5. Decision Tree Results

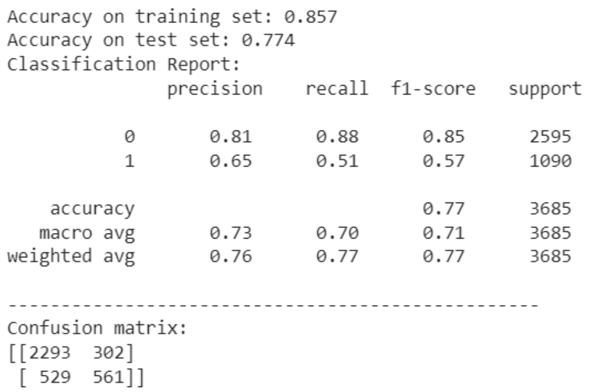
A Decision Tree model has had its hyperparameters tuned, and I have assessed the model's performance using both training and test sets. The best-tuned model is then utilized for evaluation. Additionally, feature importance visualization is done and the Decision Tree structure is exported for further analysis.



7. Decision Tree Feature Importance

C. Random Forest

During training, many decision trees are constructed using the Random Forest ensemble learning technique, which then combines them to provide a forecast that is more reliable and accurate. It is an expansion on the bagging concept and creates a diversified collection of trees using a process known as "Bootstrap Aggregating". For problems involving regression and classification, Random Forests are frequently utilized. Using bootstrap samples, a random subset of the training data is chosen to create several decision trees using Random Forest. A random subset of characteristics is taken into consideration for splitting at each node of the tree rather than all of the features. This provides variation among the trees.



```
Accuracy on training set: 0.857
Accuracy on test set: 0.774
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.88      0.85      2595
           1       0.65      0.51      0.57      1090

    accuracy                           0.77      3685
   macro avg       0.73      0.70      0.71      3685
weighted avg       0.76      0.77      0.77      3685

--------------------------------------------------
Confusion matrix:
[[2293  302]
 [ 529  561]]
```

8. Random Forest Results

In classification tasks, a majority vote among the trees determines the final prediction. The average of the predictions made by each individual tree is the final prediction for regression problems. When it comes to overfitting, Random Forests are less vulnerable than individual decision trees. They are adept at handling outliers and missing values. They offer a fair distribution of volatility and bias. Because of their strength and adaptability, Random Forests are often used in a variety of machine learning applications. When compared to individual decision trees, they frequently provide strong performance and are less responsive to hyper parameter adjustment. The hyperparameters considered for tuning and their respective ranges are as follows:

- Number of Estimators (n_estimators): 1 to 100 (in increments of 10)
- Maximum Depth (max_depth): 1 to 50
- Minimum Samples Leaf (min_samples_leaf): 1 to 60
- Minimum Samples Split (min_samples_split): 2 to 50
- Maximum Features (max_features): 1 to the number of features in the training set
- Criterion for Splitting (criterion): 'entropy' and 'gini'

A Random Forest Classifier is initialized (rf_model), and a randomized search is performed over the specified hyperparameter ranges using RandomizedSearchCV. The best-tuned hyperparameters are extracted from the search results, including the number of estimators, criterion, maximum features, minimum samples leaf, maximum depth, and minimum samples split.

The results of the randomized search are as follows:

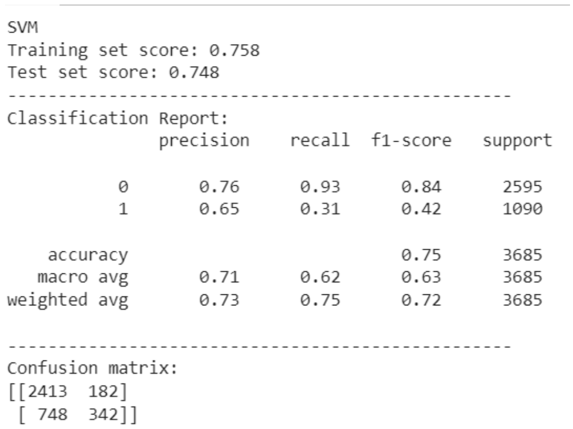
- Best number of estimators: 71
- Best criterion: entropy
- Best max features: 7
- Best min samples leaf: 11
- Best max depth: 36
- Best min samples split: 20
- Best Score: 0.788

A new Random Forest model (rf_model) is then instantiated with the best-tuned hyperparameters, and it is trained on the training set. The performance of the model is evaluated on both the training and test sets.

- Accuracy on the training set: 0.857
- Accuracy on the test set: 0.774

D. Support Vector Machines

For both classification and regression applications, Support Vector Machines (SVM) is a potent supervised machine learning technique. SVMs are frequently employed for applications like text classification, handwriting recognition, and picture classification because they perform especially well in high-dimensional domains. SVM looks for the hyperplane that divides the data into classes the best. This hyperplane is selected for a binary classification task in order to maximize the margin—that is, the distance between the hyperplane and the closest data point from either class. Support Vectors are those data points that are closest to the line where decisions are made. The hyperplane's orientation and location are affected by these points. The input properties of SVM may be transformed into a higher-dimensional space using a kernel function. 'Linear', 'poly' (polynomial), 'rbf' (radial basis function), and 'sigmoid' are examples of common kernel functions. The kind of data determines the kernel to use.



```
SVM
Training set score: 0.758
Test set score: 0.748
-------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.93      0.84      2595
           1       0.65      0.31      0.42      1090

    accuracy                           0.75      3685
   macro avg       0.71      0.62      0.63      3685
weighted avg       0.73      0.75      0.72      3685

-------------------------------------------------
Confusion matrix:
[[2413  182]
 [ 748  342]]
```

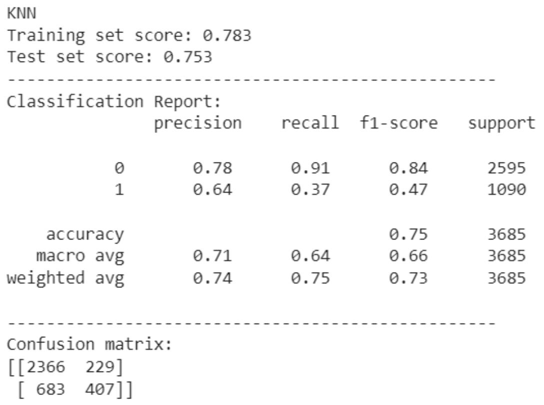
9. Support Vector Machines Results

SVMs work well in spaces with several dimensions. When there are more dimensions than samples, they function effectively. SVMs can handle complicated relationships in the data thanks to the kernel technique. SVMs are strong and flexible, although careful hyperparameter tweaking and kernel function selection may be necessary. They work especially effectively in situations when the decision boundary is not a straightforward linear plane. The hyperparameter grid is defined as {'C': [0.1, 1, 10, 100, 1000]}.

The regularization parameter C controls the trade-off between achieving a low training error and a low testing error. RandomizedSearchCV, a hyperparameter optimization technique, is employed to perform a randomized search over the specified hyperparameter grid using 10-fold cross-validation. The tuned SVM hyperparameter is found to be {'C': 10} with a corresponding best cross-validated accuracy score of approximately 75.76%. A new SVM model is instantiated using the best-tuned hyperparameter (C=10). The model is trained on the training set, and its performance is evaluated on both the training and test datasets. A detailed classification report and confusion matrix provide further insights into the model's performance.

E. K NEAREST NEIGHBOURS

A straightforward and efficient method for both classification and regression problems is K-Nearest Neighbors (KNN). This kind of instance-based learning involves the algorithm predicting based on the average of the k-nearest data points in the feature space or the majority class. Although KNN is a simple method, its effectiveness might vary depending on the number of neighbors and the chosen distance metric. It is frequently employed as a reference model to compare with more intricate algorithms.



```
KNN
Training set score: 0.783
Test set score: 0.753
--------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.91      0.84      2595
           1       0.64      0.37      0.47      1090

    accuracy                           0.75      3685
   macro avg       0.71      0.64      0.66      3685
weighted avg       0.74      0.75      0.73      3685

--------------------------------------------------
Confusion matrix:
[[2366  229]
 [ 683  407]]
```

10. K NEAREST NEIGHBOURS Results

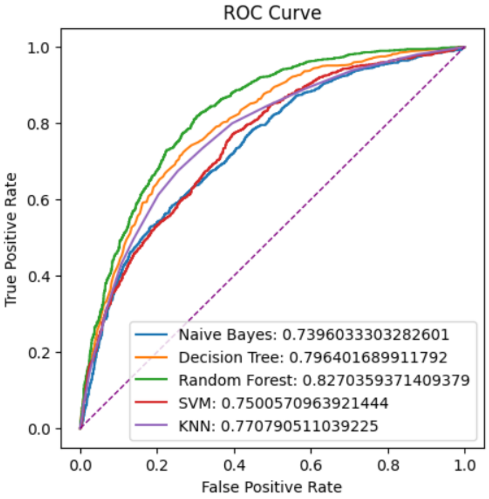
A hyperparameter grid is defined to explore values for n_neighbors ranging from 1 to 25 and p ranging from 1 to 25. This grid is used in a randomized search to find the combination of hyperparameters that maximizes accuracy. RandomizedSearchCV, a hyperparameter optimization technique, is employed to perform a randomized search over the specified hyperparameter grid using 10-fold cross-validation.

The tuned KNN hyperparameters are found to be {'p': 2, 'n_neighbors': 23} with a corresponding best cross-validated accuracy score of approximately 76.5%. A new KNN model is instantiated using the best-tuned hyperparameter {'p': 2, 'n_neighbors': 23}. The model is trained on the training set, and its performance is evaluated on both the training and test datasets. A detailed classification report and confusion matrix provide further insights into the model's performance.

2. Evaluation Metrics:

We analyze the Area Under the Curve (AUC) values and Receiver Operating Characteristic (ROC) curves for several classifiers. The trade-off between true positive rate (sensitivity) and false positive rate (specificity) at various thresholds is represented graphically by the ROC curve. AUC offers a solitary statistic for evaluating the overall performance of the classifiers.

We generated ROC curves for Naive Bayes, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN), which are the five distinct classifiers. Using their expected probabilities on the test set, the false positive rates (fpr) and true positive rates (tpr) are calculated for each classifier.Next, the AUC values for each classifier are labeled on a single graph on which the ROC curves are shown.



11. ROC Curve

The ROC curves are shown in a single graph, making it possible to compare the classifiers' discriminating powers visually. A random classifier is represented by the purple dotted line; performance that is better than random is indicated by the curves above this line.

The classifiers' capacity to discern between positive and negative occurrences is quantified by the AUC values. Better overall performance is often indicated by higher AUC values.

AUC values and ROC curve analysis provide important information on how well each classifier discriminates. With an AUC score of 0.8270, the Random Forest classifier performs better than the other one in differentiating between the two classes. AUC values for Decision Tree, KNN, SVM, and Naive Bayes range from 0.7396 to 0.7964, indicating competitive performance as well.

When choosing a classifier for a particular job, this information is essential for making an informed selection. The best appropriate model for a specific problem may be chosen with the help of the ROC curve and AUC values, which offer a thorough overview of the classifiers' performance. Depending on the particular needs and limitations of the application, more investigation and refinement may be possible.

Experimental Results

A. K-Nearest Neighbors (KNN) Classifier:

Tuned Parameters:

- p: 2
- n_neighbors: 23

Performance:

- Training Set Accuracy: 78.3%
- Test Set Accuracy: 75.3%

Classification Report:

- Precision: 0.74, Recall: 0.75, F1-Score: 0.73
- Confusion Matrix: [[2366, 229], [683, 407]]

Conclusion:

- The KNN model achieved reasonable accuracy, but there is room for improvement, especially in recall for class 1.

B. Support Vector Machine (SVM) Classifier:

Tuned Parameters:

- C: 10

Performance:

- Training Set Accuracy: 75.8%
- Test Set Accuracy: 74.8%

Classification Report:

- Precision: 0.73, Recall: 0.75, F1-Score: 0.72
- Confusion Matrix: [[2413, 182], [748, 342]]

Conclusion:

- The SVM model shows comparable performance with balanced precision and recall, indicating a robust but not highly accurate classifier.

C. Random Forest Classifier:

Tuned Parameters:

- criterion: entropy
- max_features: 7
- min_samples_leaf: 11
- max_depth: 36
- min_samples_split: 20

Performance:

- Training Set Accuracy: 85.7%
- Test Set Accuracy: 77.4%

Classification Report:

- Precision: 0.76, Recall: 0.77, F1-Score: 0.77
- Confusion Matrix: [[2293, 302], [529, 561]]

Conclusion:

- The Random Forest model demonstrates high accuracy and balanced precision and recall, indicating robust performance.

D. Decision Tree Classifier:

Tuned Parameters:

- criterion: entropy
- max_features: 7
- min_samples_leaf: 59
- max_depth: 39
- min_samples_split: 23

Performance:

- Training Set Accuracy: 79.3%
- Test Set Accuracy: 76.6%

Classification Report:

- Precision: 0.76, Recall: 0.77, F1-Score: 0.76
- Confusion Matrix: [[2247, 348], [515, 575]]

Conclusion:

- The Decision Tree model shows good accuracy and balanced performance, although slightly lower than the Random Forest.

E. Naive Bayes Classifier:

Tuned Parameters:

- var_smoothing: 0.0658

Performance:

- Training Set Accuracy: 76.1%
- Test Set Accuracy: 74.6%

Classification Report:

- Precision: 0.73, Recall: 0.75, F1-Score: 0.72
- Confusion Matrix: [[2385, 210], [727, 363]]

Conclusion:

- The Naive Bayes model provides a reasonable performance, with balanced precision and recall.

F. AUC Values:

- Naive Bayes: 0.7396
- Decision Tree: 0.7964
- Random Forest: 0.8270
- SVM: 0.7500
- KNN: 0.7708

Conclusion

- The Random Forest classifier proved to be the most appropriate for this classification challenge, as seen by its best accuracy and AUC.
- KNN and Decision Tree both shown competitive performance.
- SVM showed balanced recall and precision, however it was less accurate than Random Forest.
- Though it performed very well, Naive Bayes' accuracy and AUC were somewhat lower than those of the other models.

Recommendations

- The Random Forest model may benefit from additional optimization and fine-tuning to improve performance.
- The particular needs and trade-offs between precision, recall, and total accuracy determine which classifier is best.
- For this challenge, ensemble approaches like as Random Forest work well, demonstrating the value of using
  nmerous decision models to get reliable predictions.

Referrences

1. https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/data

2. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

3. https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset/

4. https://towardsdatascience.com/hyperparameter-tuning-for-machine-learning-models-1b80d783b946

5. https://scikit-learn.org/stable/model_selection.html

6. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_predict.html

7. https://scikit-learn.org/stable/modules/tree.html#classification

8. https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2

9.https://scikitlearn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

10. https://scikit-learn.org/stable/modules/cross_validation.html

11. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_predict.html

12.https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold

13.https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html#sphx-glr-auto-examples-model-selection-plot-roc-crossval-py

14. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html