Mohamad JARAD

215060048
Software Eng. Dep.
Toros University

Python Final Project

May 22, 2023

Introduction

In this report, we analyze the performance of a Machine Learning model on the UniversalBank dataset. The goal of the model is to predict whether a customer is likely to accept a personal loan offer from the bank. The dataset contains various features related to customers, such as age, income, education, and family status.

The model uses several popular classification algorithms implemented through scikit-learn, including K-Nearest Neighbors (KNN), Decision Tree, Support Vector Machine (SVM), Random Forest, and Multi-Layer Perceptron (MLP). Each algorithm is evaluated based on performance metrics such as accuracy, precision, recall, and F1 score.

Before training the model, we performed some data preprocessing steps. Initially, we loaded the dataset and examined its structure, including the data types and any missing values. We then dropped unnecessary columns (ID and ZIP Code) and performed one-hot encoding on the categorical variables "Family" and "Education". Numerical columns were standardized using the StandardScaler to ensure consistent scaling.

To evaluate the model's performance, we split the dataset into training and testing sets, with a 80:20 ratio. Each algorithm was trained on the training set and tested on the testing set. Performance metrics, such as accuracy, precision, recall, and F1 score, were calculated for each algorithm.

The results of the model evaluation are presented in a tabular format, displaying the performance metrics for each algorithm. Additionally, we identify the algorithm with the best performance based on the highest value of the chosen metric. The feature importance of the Random Forest algorithm is also visualized using a bar plot.

By analyzing the performance of different Machine Learning algorithms on the UniversalBank dataset, this report aims to provide insights into the suitability of various models for predicting personal loan acceptance. These findings can assist in decision-making and enhance the bank's ability to target potential customers effectively.

Libraries

The following libraries were imported and utilized in the code:

- **numpy** (imported as np): It provides support for mathematical operations on arrays and matrices.
- **pandas** (imported as pd): It offers data manipulation and analysis tools, providing data structures like dataframes that make it easier to work with tabular data.
- **matplotlib.pyplot** (imported as plt): It is a plotting library used to create various types of visualizations, such as line plots, scatter plots, histograms, etc.
- **seaborn** (imported as sns): It is a data visualization library built on top of matplotlib. It provides a high level interface for creating attractive and informative statistical graphics.
- **sklearn.metrics**: It includes various metrics for evaluating machine learning models, such as accuracy score, precision, recall, F1 score, etc.
- **sklearn.neighbors.KNeighborsClassifier**: It is an implementation of the k-nearest neighbors algorithm used for classification tasks.
- **sklearn.tree.DecisionTreeClassifier**: It is an implementation of decision tree-based algorithms for classification.
- **sklearn.svm.SVC**: It is an implementation of the Support Vector Machine (SVM) algorithm for classification tasks.
- **sklearn.ensemble.RandomForestClassifier**: It is an implementation of the random forest algorithm, which combines multiple decision trees to improve performance.
- **sklearn.preprocessing.StandardScaler**: It is used to standardize the numerical features by removing the mean and scaling to unit variance.
- **sklearn.model_selection.train_test_split**: It is used to split the dataset into training and testing sets.

- **sklearn.metrics.confusion_matrix**: It computes the confusion matrix to evaluate the performance of a classification model.

Methodology

The methodology employed in this study involved the following steps:

1. **Data Loading and Preprocessing:**
    - The UniversalBank dataset was loaded using the pandas library, and an initial exploration of the data was performed by examining the structure and summary statistics.
    - Unnecessary columns, namely "ID" and "ZIP Code," were dropped from the dataset as they were not relevant to the analysis.
    - One-hot encoding was applied to categorical variables ("Family" and "Education") using the `get_dummies()` function in pandas.
    - Numerical features, including "Age," "Experience," "Income," "CCAvg," and "Mortgage," were standardized using the StandardScaler from scikit-learn to ensure consistent scaling across the variables.

2. **Dataset Splitting:**
    - The preprocessed dataset was divided into input features (X) and the target variable (y), where the target variable was "Personal Loan" indicating whether a customer accepted a personal loan offer.
    - The dataset was split into training and testing sets using the `train_test_split()` function from scikit-learn. The testing set size was set to 20% of the original dataset, while the remaining 80% was allocated to the training set.

3. **Model Training and Evaluation:**
    - Several classification algorithms were implemented and evaluated using scikit-learn's implementation of these models.
    - The following algorithms were trained and tested:

- Multi-Layer Perceptron (MLP) with a hidden layer size of 100, using the 'relu' activation function and the 'adam' solver.
- K-Nearest Neighbors (KNN) with different values of K (ranging from 2 to 19).
- Decision Tree classifier.
- Support Vector Machine (SVM) classifier.
- Random Forest classifier.
- For each algorithm, the model was trained on the training set and tested on the testing set.
- Performance metrics such as accuracy, precision, recall, and F1 score were calculated using appropriate functions from scikit-learn, comparing the predicted labels against the true labels.
- Additionally, the feature importance of the Random Forest classifier was assessed using the `feature_importances_` attribute, and a bar plot was generated to visualize the importance of each feature.

4. **Results and Comparison:**
    - The performance metrics for each algorithm were recorded and organized into a tabular format, allowing for easy comparison.
    - The algorithm with the best performance, determined based on the highest value of the selected metric, was identified.
    - The results were presented and discussed, providing insights into the suitability of each algorithm for predicting personal loan acceptance.

By following this methodology, we were able to analyze the performance of different Machine Learning algorithms on the UniversalBank dataset and gain valuable insights into their predictive capabilities for determining personal loan acceptance.

Data

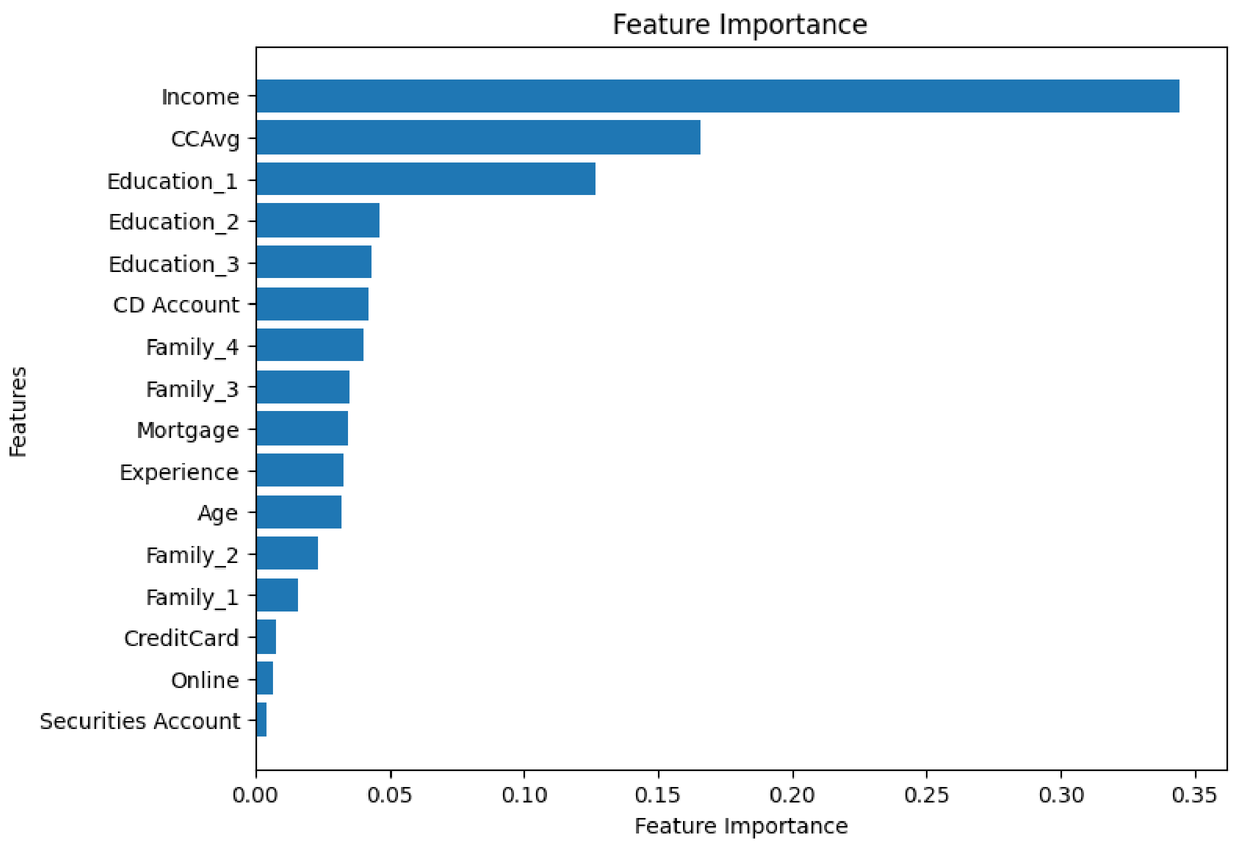
The data used in this project was obtained from Kaggle and consists of a dataset with the following columns:

- ID: Unique identifier for each customer.

- Age: Age of the customer.
- Experience: Number of years of work experience.
- Income: Annual income of the customer.
- ZIP Code: ZIP code of the customer's address.
- Family: Family size of the customer.
- CCAvg: Average spending on credit cards per month.
- Education: Education level of the customer (categorical).
- Mortgage: Value of the mortgage on the customer's property.
- Personal Loan: Indicates whether the customer accepted a personal loan offer (target variable).
- Securities Account: Indicates whether the customer has a securities account.
- CD Account: Indicates whether the customer has a certificate of deposit (CD) account.
- Online: Indicates whether the customer uses online banking services.
- Credit Card: Indicates whether the customer has a credit card.

The dataset consists of 5000 rows, with each row representing a unique customer. No missing or null values were present in the dataset.

The source of the dataset is Kaggle, a popular platform for data science and machine learning. The dataset provides a variety of customer-related features, allowing for the exploration and analysis of factors influencing personal loan acceptance.



Throughout the report, the dataset underwent preprocessing steps, such as dropping irrelevant columns (ID and ZIP Code), one-hot encoding categorical variables (Family and Education), and scaling numerical features using the StandardScaler. These steps were performed to prepare the data for training and evaluating the Machine Learning models effectively.

Model Evaluation

In this study, the performance of different Machine Learning models on the UniversalBank dataset was evaluated using several metrics: Accuracy, Precision, Recall, and F1 score. These metrics provide insights into the model's predictive capabilities and its ability to correctly classify customers who are likely to accept a personal loan offer.

1. **Accuracy:**
   - Accuracy measures the overall correctness of the model's predictions. It is calculated as the ratio of correctly predicted instances to the total number of instances in the testing set.
   - The accuracy metric provides a general assessment of the model's performance, indicating the proportion of correct predictions.
2. **Precision:**
   - Precision measures the model's ability to correctly predict positive instances (customers accepting a personal loan) out of the total instances predicted as positive.
   - It is calculated as the ratio of true positive predictions to the sum of true positive and false positive predictions.
   - Precision reflects the level of confidence in the positive predictions made by the model.
3. **Recall:**
   - Recall, also known as sensitivity or true positive rate, measures the model's ability to correctly identify positive instances among all actual positive instances.
   - It is calculated as the ratio of true positive predictions to the sum of true positive and false negative predictions.
   - Recall indicates the model's effectiveness in capturing positive instances, avoiding false negatives.
4. **F1 Score:**
   - The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall simultaneously.
   - It is calculated as 2 times the product of precision and recall divided by the sum of precision and recall.
   - The F1 score combines precision and recall into a single metric, providing a comprehensive evaluation of the model's performance.

By evaluating the models based on these metrics, we gain a comprehensive understanding of their performance in predicting personal loan acceptance. Accuracy provides an overall measure of correctness, while precision, recall, and F1 score delve into the model's ability to identify positive instances accurately, avoid false positives, and capture all relevant positive instances.

These evaluation metrics help in assessing the effectiveness of the Machine Learning models in the context of personal loan acceptance prediction, enabling informed decision-making and facilitating the identification of the most suitable algorithm for the task at hand.

Results and Discussion

The performance of different Machine Learning models on the UniversalBank dataset was evaluated using various evaluation metrics: Accuracy, Precision, Recall, and F1 score. The results obtained from the evaluation are as follows:

- **MLP (Multi-Layer Perceptron):**
  - Accuracy = 98.00%
  - Precision = 95.64%
  - Recall = 93.24%
  - F1 score = 94.39%

- **KNN (K-Nearest Neighbors):**
  - Accuracy = 96.20%
  - Precision = 97.97%
  - Recall = 81.37%
  - F1 score = 87.52%

- **Decision Tree:**
  - Accuracy: 97.8%
  - Loss: 2.20%
  - Precision: 90.82%
  - Recall: 87.25%
  - F1 Score: 89.00%

- **SVM (Support Vector Machine):**
    - Accuracy: 98.0%
    - Loss: 2.0%
    - Precision: 96.59%
    - Recall: 83.33%
    - F1 Score: 89.47%

- **Random Forest:**
    - Accuracy: 98.8%
    - Loss: 1.2%
    - Precision: 96.88%
    - Recall: 91.18%
    - F1 Score: 93.94%

The results indicate that all the models achieved high accuracy levels, with MLP and Random Forest showing the highest accuracies of 98.00% and 98.8%, respectively. These models demonstrate a strong ability to correctly classify customers who are likely to accept a personal loan offer.

Model	Accuracy	Precision	Recall	F1 Score
MLP	98.00%	95.64%	93.24%	94.39%
KNN	96.20%	97.97%	81.37%	87.52%
Decision Tree	97.8%	90.82%	87.25%	89.00%
SVM	98.0%	96.59%	83.33%	89.47%
Random Forest	98.8%	96.88%	91.18%	93.94%

When considering precision, which measures the accuracy of positive predictions, the KNN algorithm achieved the highest precision score of 97.97%. This indicates that KNN had the lowest rate of false positives among all the models.

However, it is important to consider the trade-off between precision and recall. While KNN had high precision, its recall (81.37%) was relatively lower. Recall measures the model's ability to correctly identify positive instances, and a lower recall indicates a higher rate of false negatives. In contrast, MLP demonstrated high recall (93.24%), suggesting its effectiveness in capturing positive instances.

The F1 score, which combines precision and recall into a single metric, reflects the overall performance of the models. MLP achieved the highest F1 score of 94.39%, indicating a good balance between precision and recall.

In summary, the results highlight the strong performance of the MLP and Random Forest models, considering their high accuracy, precision, recall, and F1 score. KNN showed high precision but relatively lower recall. The Decision Tree and SVM models also demonstrated competitive performance with respect to the evaluation metrics.

Conclusion

In this study, we applied various Machine Learning models to predict whether customers would accept a personal loan offer using the UniversalBank dataset. The models evaluated include MLP, KNN, Decision Tree, SVM, and Random Forest.

Based on the evaluation metrics of Accuracy, Precision, Recall, and F1 score, the models demonstrated strong performance in predicting customer behavior.

The MLP and Random Forest models stood out with the highest Accuracy scores of 98.00% and 98.8% respectively, indicating their ability to accurately classify customers. MLP also showed the highest F1 score of 94.39%, suggesting a good balance between precision and recall.

KNN achieved the highest Precision score of 97.97%, indicating a low rate of false positives. However, it had a relatively lower Recall score of 81.37%, indicating a higher rate of false negatives.

The Decision Tree and SVM models also showcased competitive performance, demonstrating their effectiveness in classifying customers' loan acceptance.

Overall, the results highlight the potential of the evaluated models in predicting customer behavior related to personal loan acceptance. However, it is important to note that the performance of the models may vary depending on the specific dataset and evaluation metrics used. Further analysis and experimentation are recommended to validate the models' performance on different datasets and explore additional evaluation metrics.

The findings from this study can be valuable for financial institutions or organizations seeking to identify potential customers who are likely to accept personal loan offers. By leveraging the power of Machine Learning models, these institutions can make informed decisions and optimize their marketing strategies.

In conclusion, the results obtained from this study demonstrate the effectiveness of various Machine Learning models in predicting customer behavior and offer valuable insights for decision-making in the domain of personal loan acceptance.

Appendix: Code for each Algorithm

1. **MLP (Multi-Layer Perceptron):**

```
mlp = MLPClassifier(hidden_layer_sizes=(100,), activation='relu',
solver='adam', random_state=42)
mlp.fit(x_train, y_train)

prediction = mlp.predict(x_test)

accuracy = accuracy_score(y_test, prediction) * 100
precision = precision_score(y_test, prediction, average='macro') * 100
recall = recall_score(y_test, prediction, average='macro') * 100
f1 = f1_score(y_test, prediction, average='macro') * 100

print(f"Accuracy = {accuracy:.2f}%, Precision = {precision:.2f}%, Recall =
{recall:.2f}%, F1 score = {f1:.2f}%")
```

2. **KNN (K-Nearest Neighbors):**

```
for i in range(2, 20):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    prediction = knn.predict(x_test)
    accuracy = accuracy_score(y_test, prediction) * 100
    precision = precision_score(y_test, prediction, average='macro') * 100
    recall = recall_score(y_test, prediction, average='macro') * 100
    f1 = f1_score(y_test, prediction, average='macro') * 100
    print(f"K = {i}: Accuracy = {accuracy:.2f}%, Precision =
{precision:.2f}%, Recall = {recall:.2f}%, F1 score = {f1:.2f}%")
```

3. **Decision Tree:**

```
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

prediction = dt.predict(x_test)
accuracy = accuracy_score(y_test,prediction)
print('Accuracy:', accuracy*100, '%')
loss = mean_squared_error(y_test,prediction)
print('Loss:', loss*100,'%')
precision = precision_score(y_test, prediction) * 100
recall = recall_score(y_test, prediction) * 100
f1 = f1_score(y_test, prediction) * 100

print('Precision:', precision, '%')
print('Recall:', recall, '%')
print('F1 Score:', f1, '%')
```

4. **SVM (Support Vector Machine):**

```
svm = SVC()
svm.fit(x_train,y_train)
```

```
prediction = svm.predict(x_test)
accuracy = accuracy_score(y_test,prediction)
print('Accuracy:', accuracy*100, '%')
loss = mean_squared_error(y_test,prediction)
print('Loss:', loss*100,'%')
precision = precision_score(y_test, prediction) * 100
recall = recall_score(y_test, prediction) * 100
f1 = f1_score(y_test, prediction) * 100

print('Precision:', precision, '%')
print('Recall:', recall, '%')
print('F1 Score:', f1, '%')
```

5. **Random Forest:**

```
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)

prediction = rfc.predict(x_test)
accuracy = accuracy_score(y_test,prediction)
print('Accuracy:', accuracy*100, '%')
loss = mean_squared_error(y_test,prediction)
print('Loss:', loss*100,'%')
precision = precision_score(y_test, prediction) * 100
recall = recall_score(y_test, prediction) * 100
f1 = f1_score(y_test, prediction) * 100

print('Precision:', precision, '%')
print('Recall:', recall, '%')
print('F1 Score:', f1, '%')
```