EECS 3401 – Introduction to AI and

Logic Programming

Instructor: Ruba Al-Omari

Project

**Loan Approval Prediction**

**Exploring and Predicting with a Dataset**

**Group Members:**

1. **Hasti Ghaneshirazi**
2. **Wenzhuo Li**
3. **Helia Ostadalipour**

**Abstract:**

This program uses machine learning and data analysis to develop models and algorithms that predict the likelihood of loan approval based on the given features. Based on the features in the dataset, the prediction of a loan approval will be conducted.

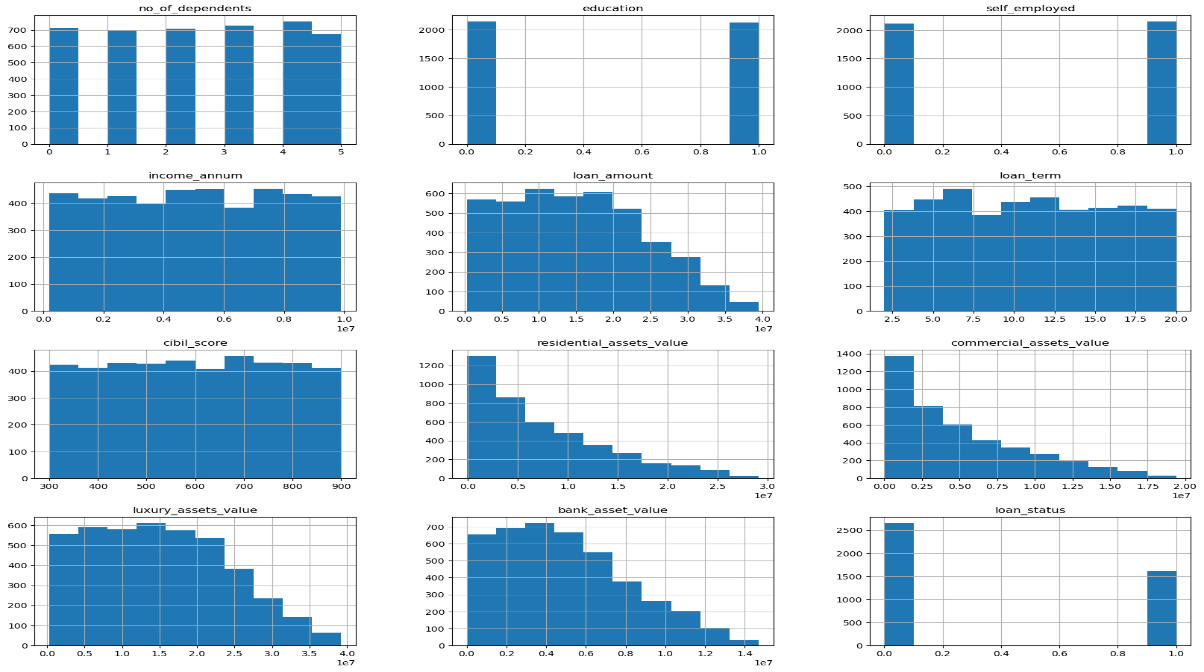
**Task 1:** Framing the problem and looking at the big picture.

* Supervised learning – We are using numerical values and the data has been labeled.
* Classification task – Predict the classes (labels) based on calculation and determining the approval of the loan (can be considered as 0 and 1)
* Batch learning
  + Small data set
  + No continuous flow of data coming into the system.
  + No need to adjust to changing data rapidly.

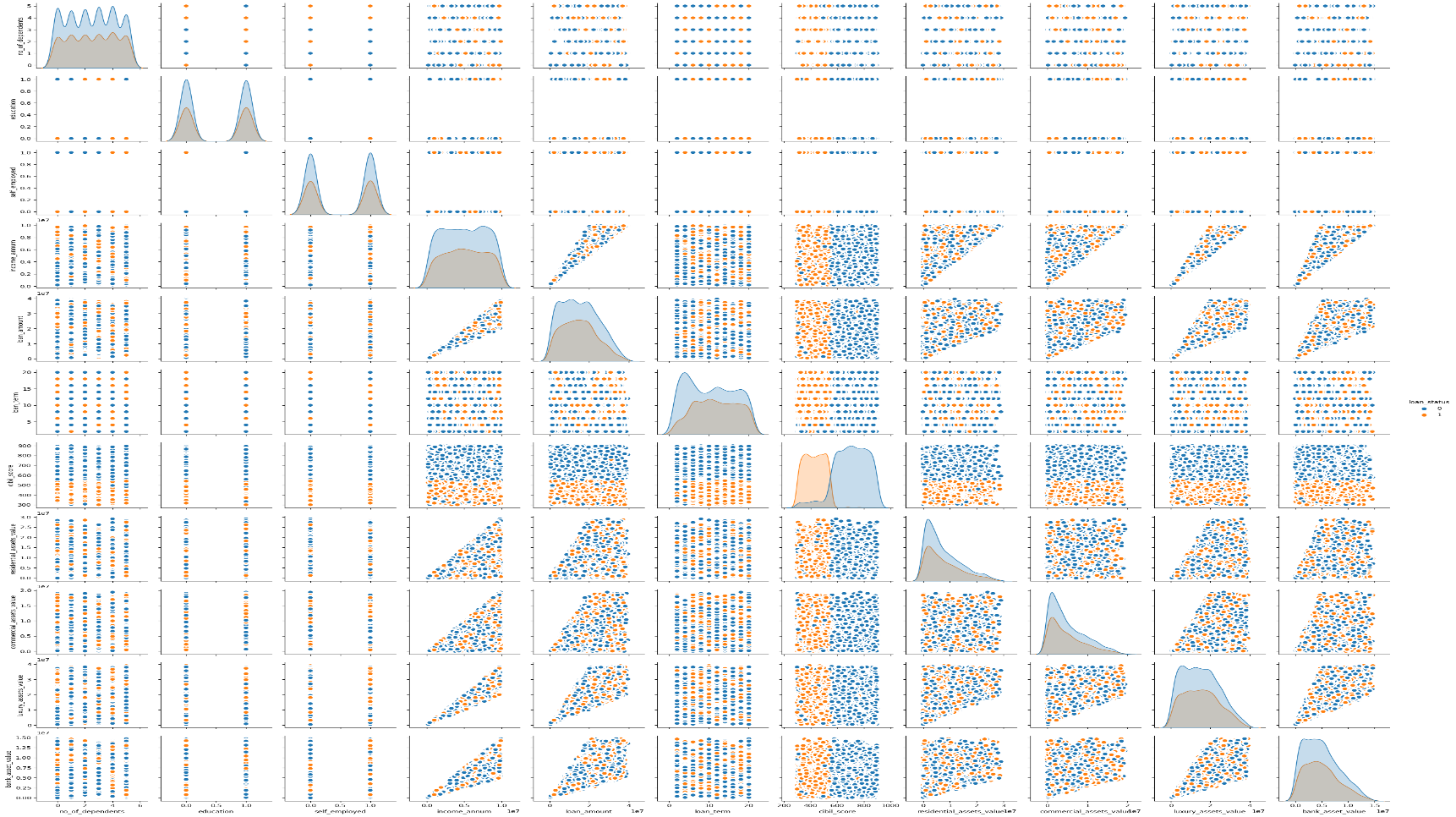
**Task 2:** A description of the dataset and 3 graphs of EDA.

In our EDAs (especially in the Correlation Matrix) we are showing that the features in our dataset don’t have a strong correlation to each other. There are a few noticeable values. First is the relation between cibil\_score and the loan\_status (target) is the reverse relationship with a correlation of -0.77. There is no other significant correlation with the target value, however, we can observe that the amount of annual\_income affects the loan\_amount and the luxury\_asset\_value directly (logical relationship). This pattern is also visible in the other visuals. For example, in the pair plot, the most distinct separated values are the ones with the highest correlations.

Histogram:



Pair Plot (scatter):



Correlation Matrix:

A screenshot of a graph

Description automatically generated

**Task 3:** Data cleaning and preprocessing.

In this step, we have cleaned the dataset so it can be trained with our chosen training algorithms. Below, are the steps on how the cleaning took order:

1. Based on our dataset and looking at our features, the first column (“loan\_id") doesn’t have a significant value in the dataset. Hence, we can drop the column.
2. Using label\_encoder function from sklearn.preprocessing library to encode the categorial target column (loan\_status being Approved/Rejected) to 0 indicating Approved (True), and 1 indicating Rejected (False).
3. We also used label\_encoder to change the rest of the object type values to numerical for ease of processing and to find better correlation between the values

**Task 4:** Training and evaluation of three machine learning algorithms, analyze findings, and compare results.

**Preprocessing:**

First, we created a pipeline using the numerical and categorical columns. In the Pipeline is:

1. Fill in the missing numerical values with the mean using a SimpleImputer

2. Scale the numerical columns using StandardScaler.

3. Fill in the missing categorical values with the most\_frequent value using SimpleImputer

4. Encode the categorical columns using OneHotEncoder

A screenshot of a computer program

Description automatically generated

The pipeline for the preprocessed dataset

After creating the preprocessed dataset, we will apply the pipeline to the dataset. After this step, the data is ready to be trained.

We have also divided our dataset into training, validation, and test sets (60-20-20)

\*Our dataset is small and the number of instances in the validation and training sets is not much. In these cases, we can skip the validation set and do an 80-20 for the training and test sets. However, for the sake of following the common practice pattern, we also considered the validation set.

**Training:**

The training of our prediction can be done by regression models. For our prediction, we are training our dataset based on the logistic regression model, random forest model, and XGBoost model.

1. Logistic regression model: we chose this model because of the properties of our dataset. Our target can be considered a binary problem with two outcomes (in our case, approved or rejected). Moreover, logistic regression is efficient with small datasets.
2. Random Forest Model: Random Forest is commonly used for classification tasks, where the goal is to assign a label or category to input data. Unlike the regular decision trees that are prone to overfitting, random forests can handle noises or outliers.
3. XGBoost: which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT). We can use this model since it’s based on shallow decision trees and also doing gradient descent in the meantime to get the best result.

**Analyzing the results:**

In our problem accuracy is more important than the other metrics. We want to know the eligible people for a loan to get approved. Since we are doing classification, we would care about our types of errors. In addition to True Positives and True Negatives, we care about False positives more than False negatives. False Negative means, we are rejecting people who are eligible to get a loan and pay the loan back in due time. However, False Positive means that we are giving loans to people who are not eligible. If eligible people get rejected from a loan, they can try again, but if people who don’t have the means to pay back the loan get approved, that will cause problems.

Hence, between our training models, we choose a model with the most accuracy (because less FP is more important to us) and compare the result with the performance of the classification. Between our training models, the Random Forest model had the best overall performance (by looking at the metrics and the confusion matrix). Here is a more specific analysis of the three training models:

1. Logistic regression: the result of our accuracy and other metrics are worse than the other training models. This is mostly because in our problem there is no clear line for the decision boundary and different combinations of all features are determining the probability of loan approval. So, the is no clear distinction between the FP/FN from TP/TN.
2. Random forest: we chose this model as our best model because of the nature of the model and algorithm as well as consideration of the results. Since the correlation between the features is not that significant except for a few numbers, we can conduct random decision trees and form the forest. Our accuracy is very high in this model and as we wanted, the FP is the least.
3. XGBoost: This is also a good training model for our dataset. The results are also almost as good as random forest. However, even though XGBoost is a powerful model, for simpler and smaller datasets like ours it’s taxing on the system. This model requires computational resources and it’s more complex for hyperparameter tuning.

In conclusion, based on our dataset with the assumption that we are not adding external data or our data is not changing internally, random forest is the best model. However, for bigger datasets or in our case more important clients with more sensitive outcomes (like corporates or people with important status) it’s better to use XGBoost since it can use Gradient boosting and will do the calculations in more detail.

**Task 5:** 3 graphs for the best performing algorithm.

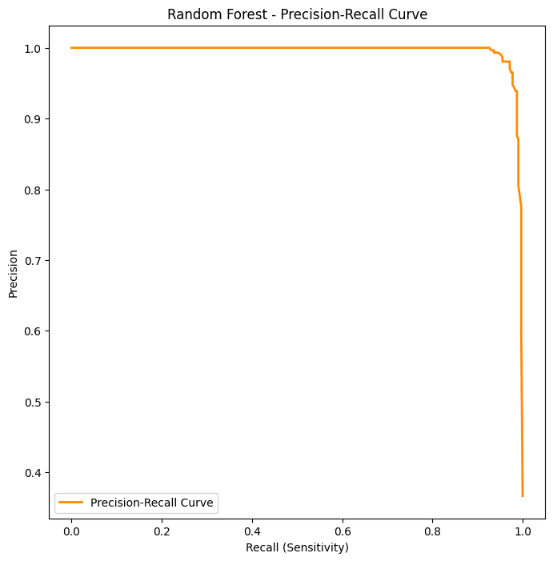
In the 3 graphs below, we show three different evaluations. First, we are seeing the feature importance of our model after the training which indicates that our assumptions and correlation were correct preprocessing. Second, []. Last, we have the ROC curve which shows the TPR vs. FPR at different classification thresholds.

Feature Importance in descending order:

A screen shot of a graph

Description automatically generated

Precision-Recall Curve



ROC Curve:

A graph of a curve

Description automatically generated

**Task 6:** Any limitations you have run into.

1. One of our main limitations for determining the main features that have the biggest correlation to the target. After seeing the values in the correlation matrix, it was a hard task to choose the features that we wanted to work with.
2. Deciding the use label\_encoding before the preprocessing to use correlation matrix or using OneHotEncode method within the preprocessing.
3. Right now that our dataset is small, we can use random forest over XGBoost. But, when our dataset gets bigger and more populated, the random forest has some shortcomings such as computational complexity for all the decision trees in the forest, bias toward dominant classes, and hyperparameter sensitivity.
4. Choosing appropriate plots for the result was a challenge since most of the graphs were for regression models or did not work well with a small dataset.

**Appendix 1:**

**Task 7:** Appendix 1: Source code with proper comments and attribution to any code you have reused.

**Task 8:** Appendix 2:

* Link to your dataset.
* Link to your executed Jupyter Notebook on github. The notebook should contain the code for your machine learning models and should show the results. Your code should be properly commented, and you must attribute any code you are using from someone else.