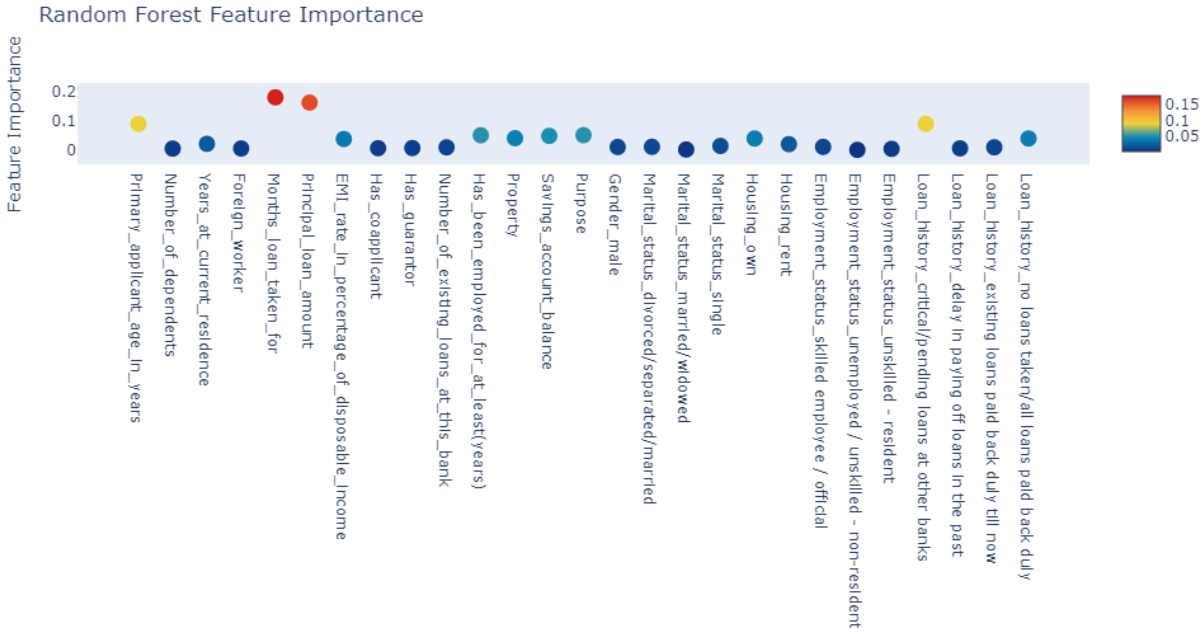
Deepraj Mazumder Assignment:

Task 1 Documentation:

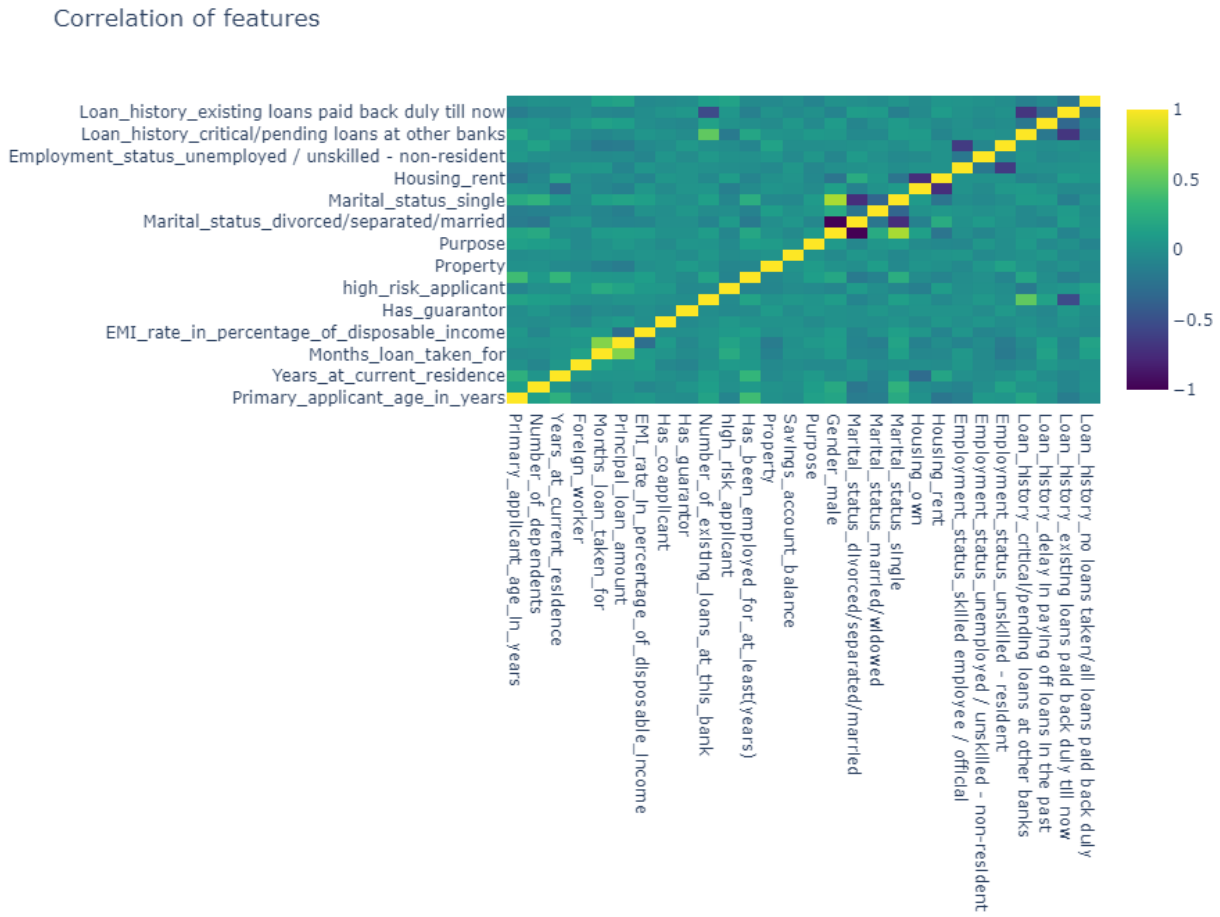
Click here for the notebook of Task 1

The given data can be divided into various segments to understand which one causes the highest variance in the target variable. Some of them are the applicants' personal status (For that, columns like Marital_status (string), Gender (string) must be considered), financial position (Housing (string), Property (string)) and both of these segments fall under the applicants' status. I divided the columns based on category and numerical dtypes and then handled the data accordingly.

Now for understanding which features, I used the random forest and then plotted interactive diagrams using plotly.

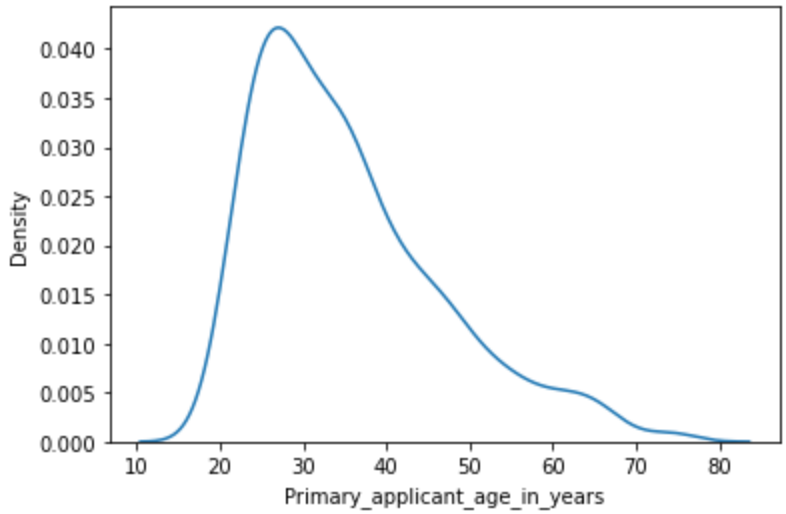


From here it is visible that the loan amount and the lone durations are the most important features and this is self-explanatory too. Now to understand which segments have high-Thi applicants, I plotted the heat map:



Note: The above two graphs are dynamic, so we need to run the cells each time to see the output.

This image is static but in the jupyter notebook, it is visible that as age increases, the risk of the applicant reduces. It looks like younger people are more likely to be a risk but the data is right skewed so the number of younger applicants is high in general.



The dependence of the applicant's personal info on the target variable is visible here:

Click [here]for the link to the tableau dashboard

The rest of the complete analysis of the data and explanation are given in the data_insights.ipynb file.

Task 2 Documentation:

Click [here] for the notebook of task 2.

Before going directly to the model testing, we made the data ready for feeding to the model.

- To begin with, we handled the unbalanced nature of the target variable using the over_sampling method, as undersampling is never a good option, especially when we have 1000 rows of data.

```
In [17]: sampler = SMOTEN(random_state=0)
         X_res, y_res = sampler.fit_resample(x, y)
         print(f"Class counts after resampling {Counter(y_res)}")

         Class counts after resampling Counter({0.0: 700, 1.0: 700})
```

- Now that we have balanced data, we do the standardization so that we can get the best use of the dimensionality reduction process.
- While training the models, I used the tree-based models and their boosting models this is because the data has less randomness and this is why we did not apply transformations to convert the data to normally distributed.

Note: the notebooks loo better in jupyter Notebooks.

Answer to the questions asked:

1. **Explain your intuition behind the features used for modelling.**

Answer: The main motive that I had while selecting the features is having efficient data for the model. This is why at the start of the process I wanted to keep just 15-20 features out of 27 features just to reduce the dimension. For that, I checked which features are highly related to the target variable and then understood that the features which are directly connected to the applicant's application for a loan (like the duration of loan etc) are the most important features.

2. **Are you creating new derived features? If yes explain the intuition behind them.**

Answer: I did not create features manually but rather used PCA to create features out of the existing features. This is because models perform really well with data with fewer features and also it is a bad idea to keep features that do not explain much variance. Such features only act as a spoiler to the model.

3. **Are there missing values? If yes how do you plan to handle it**?

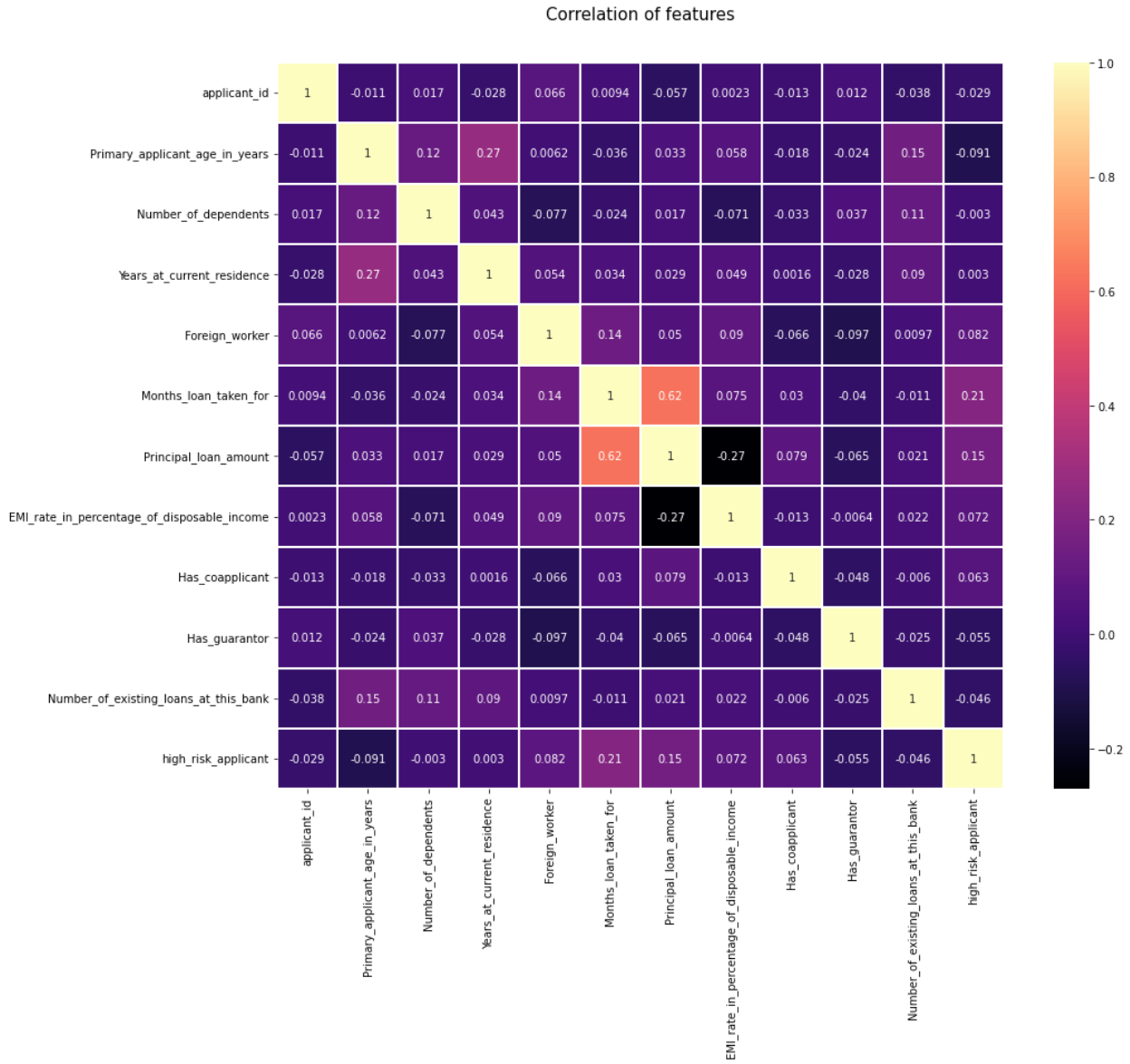
Answer: There are many missing values, especially in the categorical columns. I dropped the columns that have missing values greater than 20% and the rest I planned to impute it. As that is the most accurate way to fill missing values. A few weeks back I read a blog of AnalyticsVidya which is a wonderful resource for handling missing values. Click here for the blog.

4. **How categorical features are handled for modelling?**

Answer: We encode the categorical features using various encoders. If the encoder has a relation among them then we use ordinal encoding and if there is no relation then we use one hot encoding.

5. **Describe the correlation of the features using a correlation matrix. Tell us about a few correlated features & share your understanding of why they are correlated.**

Answer: Here is the picture to understand the correlation of our dataset:



Here we can see that Principal_loan_amount and Months_loan_taken for are 62% correlated, which means the higher the loan amount, the more time the applicant asks for repayment and that makes sense.
'Principal loan amount' and 'EMI_rate_in_percentage_of_disposable_income' are negatively correlated which means the applicant's total outgo on EMIs does not exceed a certain percentage limit of your take-home salary.
The High_risk_applicant column at the bottom also shows its correlation with other variables.

6. **Do you plan to drop the correlated feature? If yes then how.**

Answer: We need the features for filling the null values but after that, we perform PCA which takes care of the correlated features.

7. **Which ML algorithm do you plan to use for modelling?**

Answer: I used the Random forest model and the ADAboost model. This is because the logistic regression (linear models) will not perform better than the tree-based models with less randomness.

8. **Train two (at least) ML models to predict the credit risk & provide the confusion matrix for each model.**

Answer: The confusion matrices are as follows:

Random Forest:

```
[[103  38]
 [ 39 100]]
              precision    recall  f1-score   support

         0.0       0.73      0.73      0.73       141
         1.0       0.72      0.72      0.72       139

    accuracy                           0.73       280
   macro avg       0.72      0.72      0.72       280
weighted avg       0.72      0.72      0.72       280
```

AdaBoost:

```
[[ 88  53]
 [ 29 110]]
              precision    recall  f1-score   support

         0.0       0.75      0.62      0.68       141
         1.0       0.67      0.79      0.73       139

    accuracy                           0.71       280
   macro avg       0.71      0.71      0.71       280
weighted avg       0.71      0.71      0.71       280
```

9. **How you will select the hyperparameters for models trained in the above step?**

Answer: I used GridSearch for selecting the best hyperparameters. I passed lists of various hyperparameter values and GridSearchCV selected the one with the best performance.

10. **Which metric(s) you will choose to select between the set of models?**

Answer: I will use the Recall score to select the models as we need the least amount of False-Negative values from a business point of view of the loan provider.

11. **Explain how you will export the trained models & deploy them for prediction in production.**

Answer: We will use pipelines for that. After creating pipelines, we use the streamlit library to create a web app and deploy it to Heroku. In the web app, we will add the columns for the user/client to enter and then our user will get the result.

Conclusion:

The dataset is well presented but getting features that are directly related to the loan application will get us a better recall and other performance measures. Such features will be normally distributed too by default.