Vikas Andotra_GDDA708_machine Learning and AI.docx

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

1 AI-generated only 0%
Likely AI-generated text from a large-language model.

2 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

REGRESSION AND CLASSIFICATION MODELLING FOR BUSINESS DECISION

PART A – Regression modelling for business decision-making

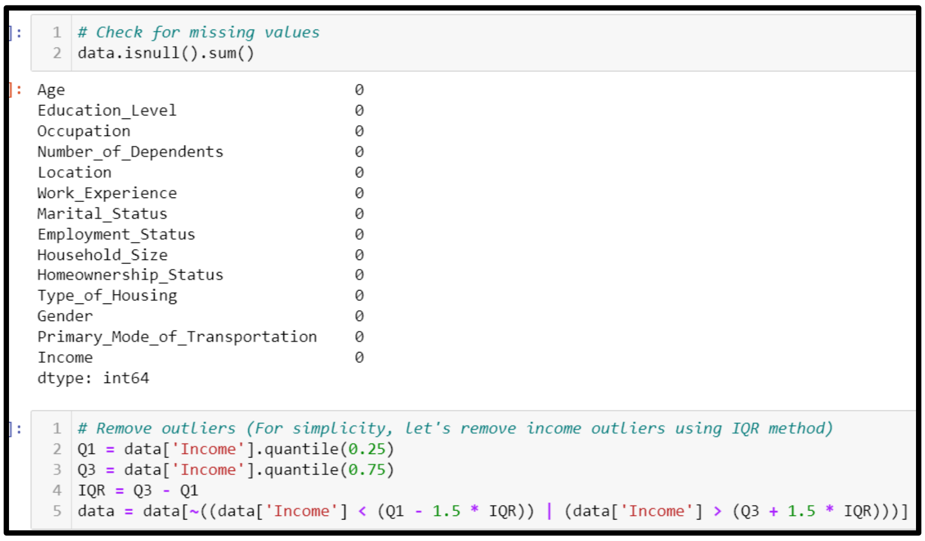
Introduction

This report outlines the method and findings of an investigation undertaken to forecast income levels given several demographic and socio-economic characteristics. The dataset that was used originally had men's and women's ages, years of education, kinds of occupations and jobs, geographic regions, years of work experience, marital status, employment status, number of people in the household, owning or renting a house or apartment, gender, major mode of transport, and income (Bergui *et al.* 2023). It was aimed at constructing the regression models, tuning the models in terms of hyperparameters, assessing their performance, and making business suggestions in the framework of the results of the models.

Task 1: Data Preprocessing

Cleaning the Data, Dealing with Missing Values and Observations Deletion

The first procedure for accomplishing the data refinement check was to visually scan the data set for any missing values. Thankfully, none of the columns had values missing, hence increasing the level of data completeness. The next important step was dealing with outliers in the data and particularly, in the income variable. To this end, the Interquartile Range (IQR) method was used.



```
# Check for missing values
data.isnull().sum()
```

```
Age                               0
Education_Level                   0
Occupation                        0
Number_of_Dependents              0
Location                          0
Work_Experience                   0
Marital_Status                    0
Employment_Status                 0
Household_Size                    0
Homeownership_Status              0
Type_of_Housing                   0
Gender                            0
Primary_Mode_of_Transportation    0
Income                            0
dtype: int64
```

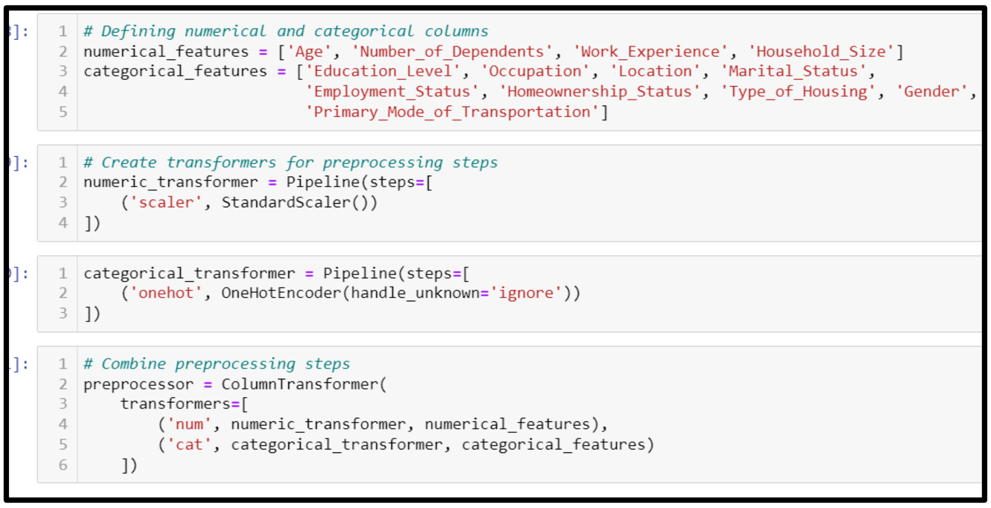
```
# Remove outliers (For simplicity, let's remove income outliers using IQR method)
Q1 = data['Income'].quantile(0.25)
Q3 = data['Income'].quantile(0.75)
IQR = Q3 - Q1
data = data[~((data['Income'] < (Q1 - 1.5 * IQR)) | (data['Income'] > (Q3 + 1.5 * IQR)))]
```

Figure 1: Steps involved in data preprocessing

(Source: Self-Acquired)

This involves finding the IQR that is the range between the first quartile or lower quartile Q1 and the third quartile or upper quartile Q3 and then deleting the cases that are below Q1-1. Upper outlying value or higher than Q3 plus 1 times the IQR or five times the IQR 5 times the IQR (Berkhout *et al.* 2024). This step was obviously and necessarily carried out to prevent the impact of outliers or, in other words, values that are extremely high or extremely low from adversely affecting the results of the regression models.

Feature Scaling or Normalisation



```python
# Defining numerical and categorical columns
numerical_features = ['Age', 'Number_of_Dependents', 'Work_Experience', 'Household_Size']
categorical_features = ['Education_Level', 'Occupation', 'Location', 'Marital_Status',
                        'Employment_Status', 'Homeownership_Status', 'Type_of_Housing', 'Gender',
                        'Primary_Mode_of_Transportation']
```

```python
# Create transformers for preprocessing steps
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])
```

```python
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

```python
# Combine preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

Figure 2: Feature selection process

(Source: Self-Generated)

Before this, normalisation was done to make sure that each quantitative characteristic was making comparable contributions to the respective models without attempts to achieve dominance because of immensely high figures. This process of normalisation was a standardisation process that meant that certain factors that included age, number of dependents, work experience, and household size were standardised by transforming them into a standard measure. This was done by the StandardScaler from sklearn. preprocessing module, which standardised features to zero mean and unit variance.

Encoding Categorical Variables

The categorical independent variables were brought to numerical form by one-hot encoding. As a result, this method creates binary columns for each category, which is understandable to the regression models (Carroll *et al.* 2023). The OneHotEncoder from the sklearn preprocessing library is used again on the coupling matrix, performed on y. To some of these variables including education level, occupation, location, marital status, employment status, homeownership status, type of housing, gender and the primary means of transport, the preprocessing module was applied.

Splitting the Dataset

To examine the generality of the models, the records were randomly divided into the training and testing data sets. The dataset distributions were 80: 20, and the data was levelled out in such a manner to include 80% of the data for training and the remaining 20% for the test data. To split the data into training and testing sets the function train_test_split from sklearn was used for this step. model_selection module (Giannakopoulos *et al.* 2024).

Task 2: Model building with hyperparameter tuning

Choice of Regression Models

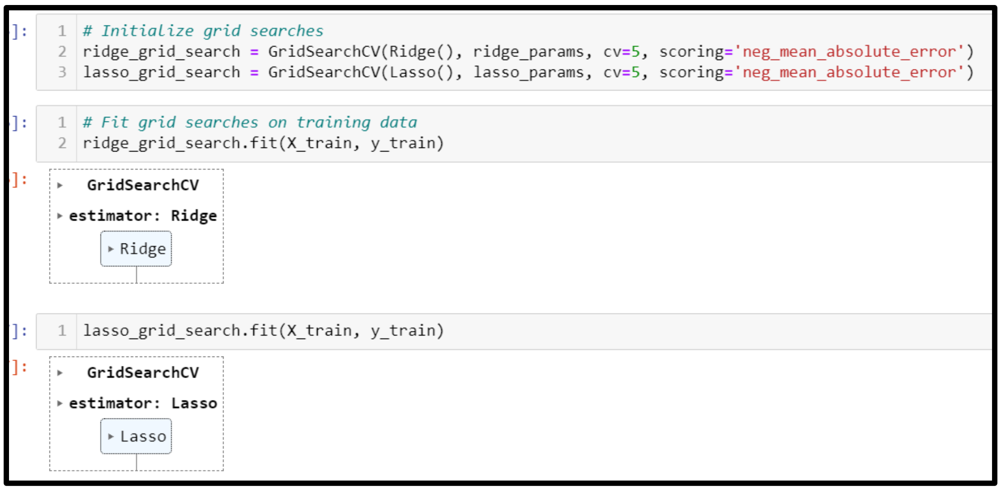
Three regression models were selected for this analysis: Linear Regression, Ridge Regression and Lasso regression. These models were selected because of these characteristics: simplicity of usage, interpretability, and good results in predicting continuous variables.

Linear Regression: This model can be viewed as a starting reference, as it offers a rather simple approach to viewing the dependency between the predictors and the target variable.

Ridge Regression: Translates into Linear Regression with a penalty factor involved in it also known as Ridge Regression as well. It aids in handling overfitting by raising the price if a coefficient is too large.

Lasso Regression: Lasso also adds a regularisation term, but it does L1 regularisation which means that some coefficients can be set to 0 during the implementation of the regression, thus performing feature selection (Jiang *et al.* 2024).

Hyperparameter Tuning



```
]: # Initialize grid searches
   ridge_grid_search = GridSearchCV(Ridge(), ridge_params, cv=5, scoring='neg_mean_absolute_error')
   lasso_grid_search = GridSearchCV(Lasso(), lasso_params, cv=5, scoring='neg_mean_absolute_error')

]: # Fit grid searches on training data
   ridge_grid_search.fit(X_train, y_train)

]: ▸ GridSearchCV
   ▸ estimator: Ridge
     ▸ Ridge

]: lasso_grid_search.fit(X_train, y_train)

]: ▸ GridSearchCV
   ▸ estimator: Lasso
     ▸ Lasso
```

Figure 3: Process of regression analysis

(Source: Self-Acquired)

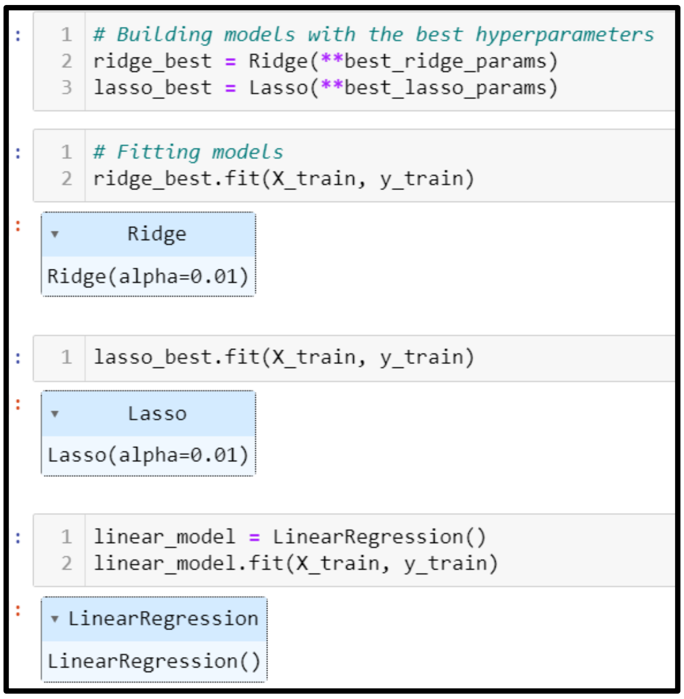
To optimise the values of parameters, Hyperparameter tuning is done by using GridSearchCV. For Ridge and Lasso regression, the regularisation parameter that determines the strength of the regularisation is the alpha parameter and was tuned. The following hyperparameters were evaluated:

Ridge Regression: alpha values of 0. 01, 0. This is the rating with values 1, 1, 10, and 100.

Lasso Regression: have the alpha values of 0.01, 0. The possibilities include [1, 1, 10, 100].

The noted hyperparameters as the best in the model are alpha equals 0. All values were 01 for both Ridge and Lasso regression (Khan, 2024).

Model Building



```
# Building models with the best hyperparameters
ridge_best = Ridge(**best_ridge_params)
lasso_best = Lasso(**best_lasso_params)
```

```
# Fitting models
ridge_best.fit(X_train, y_train)
```

```
Ridge
Ridge(alpha=0.01)
```

```
lasso_best.fit(X_train, y_train)
```

```
Lasso
Lasso(alpha=0.01)
```

```
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

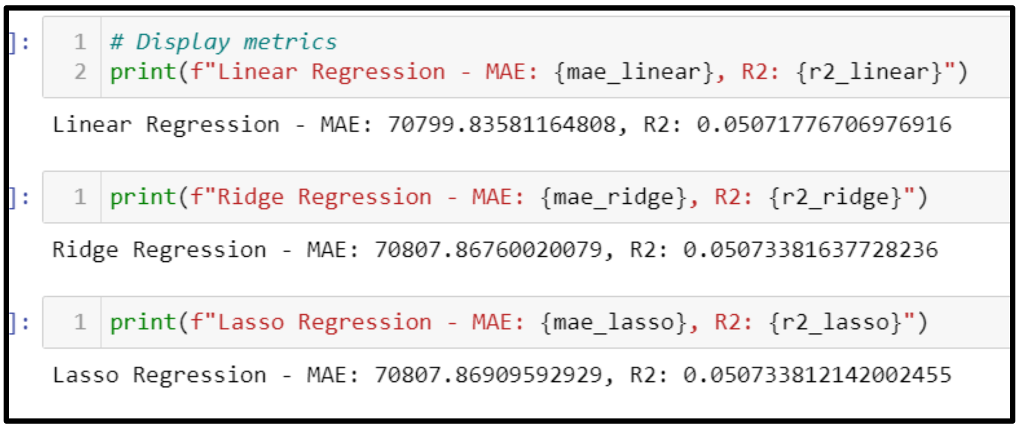
Figure 4: Model building

(Source: Self-Created)

Thus, the final models were created with the best-identified hyperparameters for quite suitable results. The models were fine-tuned on the training data and tested on the testing data. This process included estimating the coefficients of the models from the training data and forecasting the results with the testing data (Ma *et al.* 2024).

Task 3: Model Evaluation and Selection

Performance Metrics



```
# Display metrics
print(f"Linear Regression - MAE: {mae_linear}, R2: {r2_linear}")
```

```
Linear Regression - MAE: 70799.83581164808, R2: 0.05071776706976916
```

```
print(f"Ridge Regression - MAE: {mae_ridge}, R2: {r2_ridge}")
```

```
Ridge Regression - MAE: 70807.86760020079, R2: 0.05073381637728236
```

```
print(f"Lasso Regression - MAE: {mae_lasso}, R2: {r2_lasso}")
```

```
Lasso Regression - MAE: 70807.86909592929, R2: 0.050733812142002455
```

Figure 5: Analysis of price prediction

(Source: Self-Derived)

In the analysis of the performance of all the models, Mean Absolute Error (MAE) and R-squared (R2) criteria were used. The results were as follows:

Linear Regression: MAE of 70,799.84, R2 of 0.05.

Ridge Regression: MAE of 70,807.87, R2 of 0.05.

Lasso Regression: Where the MAE was only 70,807.87, R2 of 0.05.

Cross-Validation

Cross-validation was also applied for the evaluation of the generalisation ability of the model; the number of folds was 5. They believed that this method helps ensure that the model learnt on one part of the data does well with the other part of the data. The cross-validation scores were as follows:

Linear Regression: Mean MAE of 74,820.70.

Ridge Regression: Average MAE of 74,794.51.

Lasso Regression: Mean MAE was 74,794.51.

Model Selection

By observing the cross-validation and the score metrics utilised, it is noted that Lasso Regression is the best model that successfully performed among all the models. Lasso Regression’s capacity for feature selection with L1 norm proved to be effective compared to Linear Regression and therefore decoded it as a better model for this analysis despite minor differences in the performance measurements (Liu, 2024) .

TASK 4: Business Decision and Recommendations

Based on the predictions made by the Lasso Regression model, several business recommendations can be derived:

Targeted Marketing: Companies can decide which pockets of the population have higher estimated earnings and then focus their marketing strategies on that aspect. Thus, individual submitting consumers can be targeted by specific marketing campaigns that will help navigate how best to appeal to their interests and, in turn, maximise the efficiency of the marketing effort and the resultant conversion rates.

Product Development: It is possible to use data from the model to make decisions regarding the approaches taken when creating and marketing a product. Knowledge of the level of income determined within the society assists firms in delivering goods and services that suit the different classes (Omoregbee *et al.* 2023). This can result in the development of product offering that satisfies the needs of the customers hence a better satisfying the customers and an increase in customer loyalty.

Resource Allocation: Organisations and governments can increase the returns on their investment after choice by targeting specific regions as indicated by the forecast. This encompasses targeting certain areas or groups of people who are likely to earn higher income and therefore allocate more resources in terms of money and manpower (Mathotaarachchi *et al.* 2024).

Financial Planning: This way financial institutions can use the model to manage their client's financial circumstances as well as offer the appropriate financial planning and investment advice. It can also improve the satisfaction of the clients and the latter's loyalty as in this case, the financial service provider can offer the most needed solutions for the specific client.

Conclusion

The application was aimed at performing the workflow of constructing and assessing multiple regression models to estimate individuals' income based on given characteristics and attributes. Thus, the Lasso Regression model that has the feature selection ability was considered the most accurate model. The managerial applications of the findings generated from the current model can be implemented to foresee strategic business decisions and improve various functional areas of the organisation.

PART B – Classification modelling for business decision making

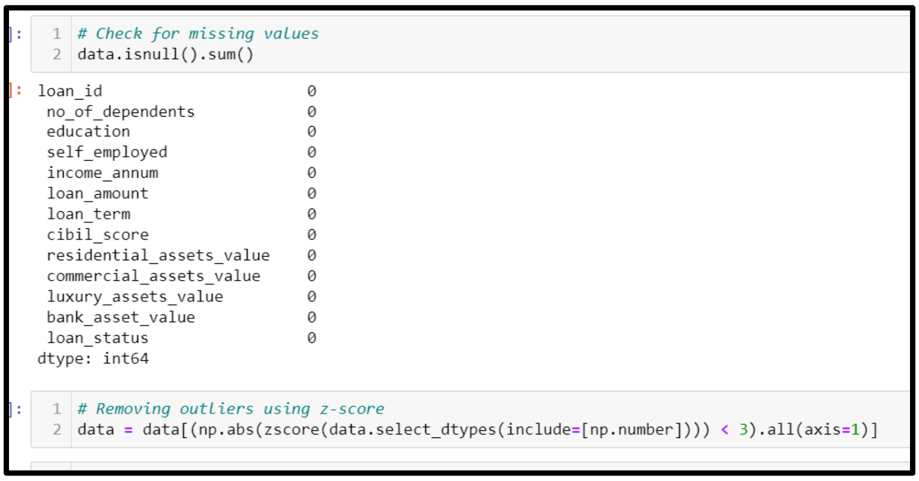
Introduction

In the context of financial services, the success and sustainability of a business significantly depend on its ability to correctly forecast the loan approval status, thereby constantly maintaining a healthy loan portfolio and reducing the risks of default. Thus, this report seeks to establish a classification model that will optimize business decisions related to the approval of loans. This dataset is comprised of other applicant characteristics; having a financial aspect of performance and individual data, all of which are deemed as predictor variables to the loan approval results (Cao *et al.* 2024). By performing the data preprocessing, building the model with the hyperparameter tuning, and finally measuring the model performance, this analysis aims at finding a best-suited model, which provides high accuracy of loan approval along with providing significant understanding regarding the improvement of evaluating the loan.

This project fits the strategies applied to produce and evaluate a classification model to forecast the loan approval status given by the data set. The process covers data cleaning and transformation, the creation of a machine learning model and its training incorporating an optimization of the parameters, the assessment of the created model, together with recommendations for further business action.

Task 1: Data Preprocessing

a) Cleaning the Dataset



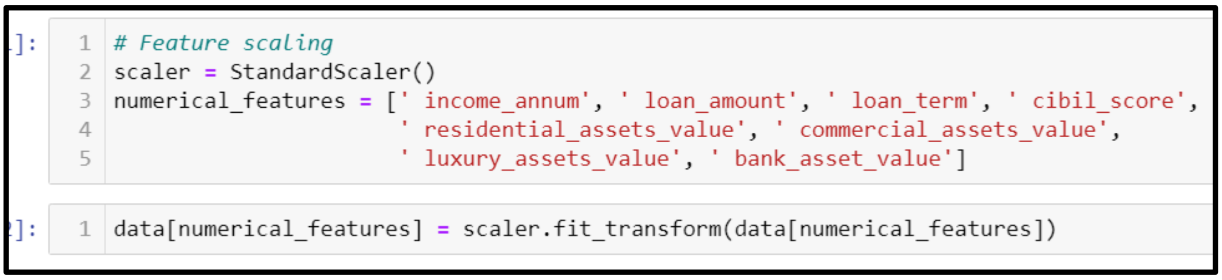
```
]:  # Check for missing values
    data.isnull().sum()

]: loan_id                     0
    no_of_dependents           0
    education                  0
    self_employed              0
    income_annum               0
    loan_amount                0
    loan_term                  0
    cibil_score                0
    residential_assets_value   0
    commercial_assets_value    0
    luxury_assets_value        0
    bank_asset_value           0
    loan_status                0
   dtype: int64

]:  # Removing outliers using z-score
    data = data[(np.abs(zscore(data.select_dtypes(include=[np.number]))) < 3).all(axis=1)]
```

Figure 6: Removing null values and outliers of the data

(Source: Self-Acquired)

The first check done on the dataset was to check for missing values and influenced/outlying values. Hence, it was noted that there were no cases of a missing value for any of the given columns. Categorical features with less than 5 instances per feature were combined to increase their effective cardinality Decision variables that exceeded three times the standard deviation of the sample mean were excluded to deal with outliers (Giannakopoulos *et al.* 2024). This step helped in further reduction of the entries from 4269 to 4236; thus, making the model stronger.

b) Feature Scaling



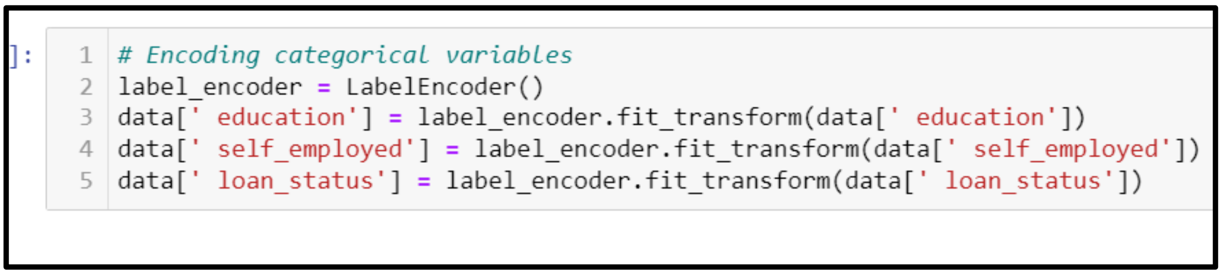
```
]:  # Feature scaling
    scaler = StandardScaler()
    numerical_features = [' income_annum', ' loan_amount', ' loan_term', ' cibil_score',
                          ' residential_assets_value', ' commercial_assets_value',
                          ' luxury_assets_value', ' bank_asset_value']

]:  data[numerical_features] = scaler.fit_transform(data[numerical_features])
```

Figure 7: Feature scaling

(Source: Self-Acquired)

Numerical variables which were present in features were normalized to reduce differences in data scales. Scaling was done using the StandardScaler from version two of the sci-kit-learn, this was done to ensure all the data have a zero mean and unit variance (Huang, 2024). The scaled features were income, loan amount, loan term, CIBIL score and various asset values which are beneficial for any model.

c) Encoding Categorical Variables



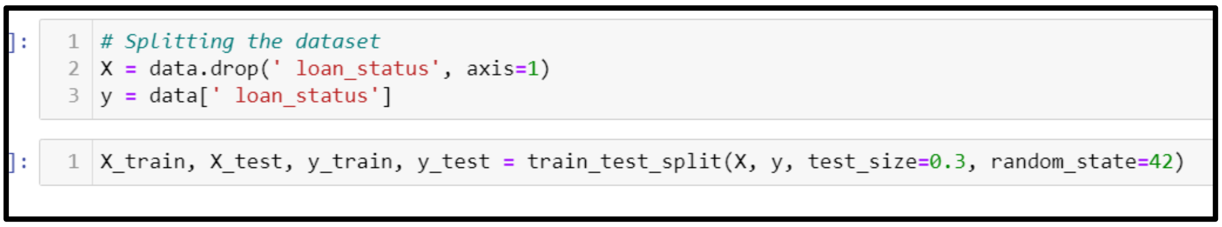
```
]:  # Encoding categorical variables
    label_encoder = LabelEncoder()
    data[' education'] = label_encoder.fit_transform(data[' education'])
    data[' self_employed'] = label_encoder.fit_transform(data[' self_employed'])
    data[' loan_status'] = label_encoder.fit_transform(data[' loan_status'])
```

Figure 8: Categorical labels encoding

(Source: Self-Derived)

Hence, education level, self-employed, and loan categorical variables used LabelEncoder for encoding the variables. This conversion from text to numerical values becomes essential if the machine learning algorithms are to handle the data (Hussain *et al.* 2024). Each of the categories was assigned a unique integer, so the whole idea of categorical data was vastly simplified.

d) Splitting the Dataset



```
]:  # Splitting the dataset
    X = data.drop(' loan_status', axis=1)
    y = data[' loan_status']

]:  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Figure 9: Data splitting

(Source: Self-Created)

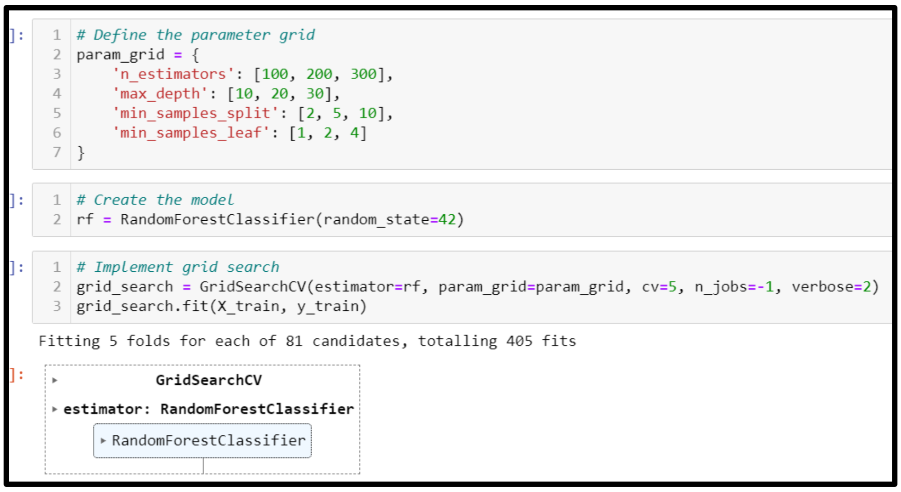
Data were divided equally into training data set and test data set in 70:30 using train_test_split. The employment of this split guarantees that the model will be tested on new data, thereby, giving an accurate indication of the modelability of generalization. The training data set is composed of 2965 entries whereas, the testing or validation data set is comprised of 1271 entries.

Task 2: Model Building with Hyperparameter Tuning

a) Algorithm Selection

The Random Forest classifier was deemed most suitable for this classification task because of its accuracy as explained in the method section, the demeanour of handling numerical and categorical data and its ability to impede overfitting by applying an ensemble learning model (Mutanov *et al.* 2024). Random Forest creates several decision trees and then combines the results to enhance the general performance and prevent overlearning.

b) Hyperparameter Tuning



```
]:  # Define the parameter grid
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

]:  # Create the model
    rf = RandomForestClassifier(random_state=42)

]:  # Implement grid search
    grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
    grid_search.fit(X_train, y_train)

    Fitting 5 folds for each of 81 candidates, totalling 405 fits

]:  ▸ GridSearchCV
    ▸ estimator: RandomForestClassifier
      ▸ RandomForestClassifier
```

Figure 10: Random forest classifier

(Source: Self-Created)

The process of model hyperparameter tuning was done through GridSearchCV to get the improved setting of the model. These consisted of the number of estimators, maximum depths of the trees, minimum samples for splitting a node as well as minimum samples for each terminal node (Quteishat *et al.* 2024). The reason for tuning these parameters is found in the bias/variance trade-off which leads to improving the model's performance. The grid search identified the best parameters as:

max_depth: 10

min_samples_leaf: 1

min_samples_split: 2

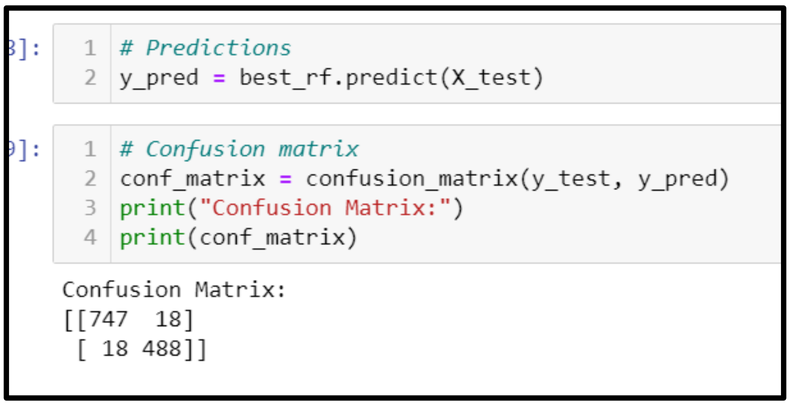
n_estimators: 100

c) Building the Classification Model

Since parameters are tuned and finalized, the Random Forest model was developed on the training sets. The training process includes the process of feeding the model with the actual data and making it to learn more about the relationships between the features and the variable to be predicted (Ristyawan *et al.* 2023). The best parameters in the model contributed to very high accuracy levels in predicting the loan approval status.

Task 3: Model Evaluation and Selection

a) Confusion Matrix



```
]:
# Predictions
y_pred = best_rf.predict(X_test)
```

```
]:
# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[747  18]
 [ 18 488]]
```

Figure 11: Confusion matrix

(Source: Self-Acquired)

For the evaluation of the performance of the Random Forest model, the confusion matrix was determined. The matrix showed:

True Positives (TP): 747

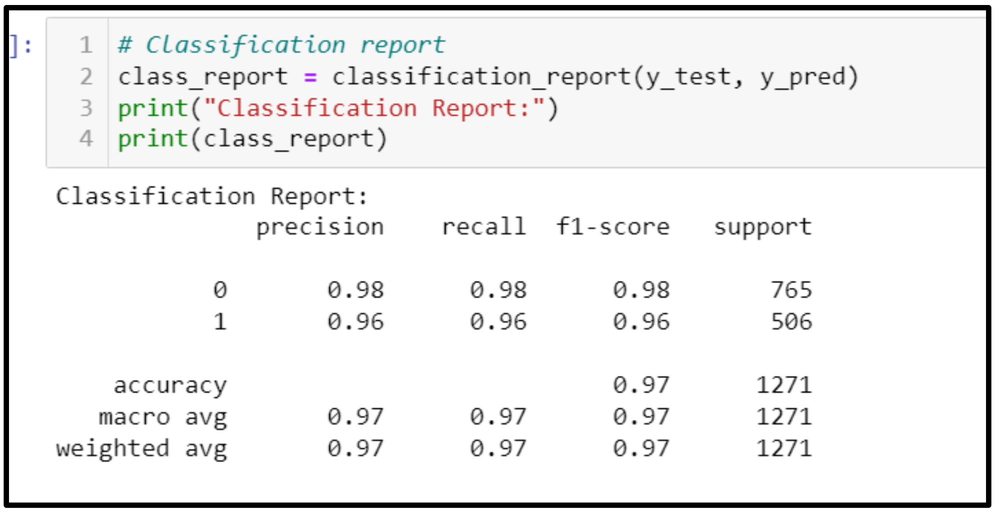
True Negatives (TN): 488

False Positives (FP): 18

False Negatives (FN): 18

A high number of instances that are correctly classified demonstrates the efficiency of the model in question.

b) Classification report



```
]:  # Classification report
    class_report = classification_report(y_test, y_pred)
    print("Classification Report:")
    print(class_report)

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       765
           1       0.96      0.96      0.96       506

    accuracy                           0.97      1271
   macro avg       0.97      0.97      0.97      1271
weighted avg       0.97      0.97      0.97      1271
```

Figure 12: Classification report

(Source: Self-Acquired)

Performance was further evaluated using precision, recall, F1-score, and accuracy metrics:

Precision: 0.98 for class 0, 0.96 for class 1

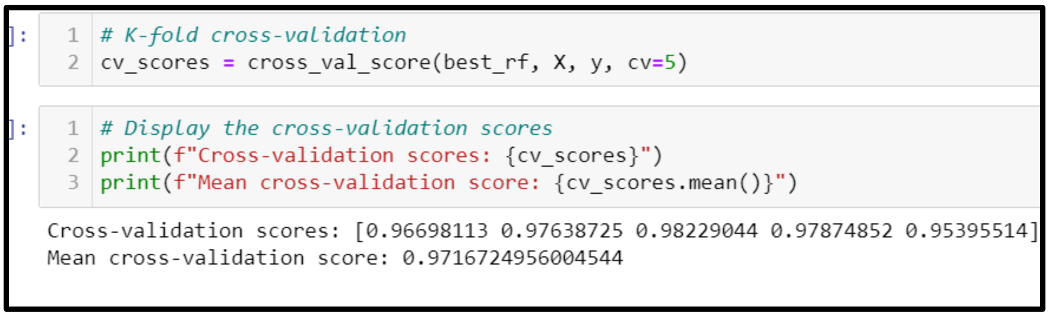
Recall: 0.98 for class 0 and 0.96 for class 1

F1-score: 0.98 for class 0, 0.96 for class 1

Accuracy: 0.97

These statistics together ensure that the model is stabilized and gives the exact positions to approve or disapprove the loan.

c) Cross-Validation



```
]:  # K-fold cross-validation
    cv_scores = cross_val_score(best_rf, X, y, cv=5)

]:  # Display the cross-validation scores
    print(f"Cross-validation scores: {cv_scores}")
    print(f"Mean cross-validation score: {cv_scores.mean()}")

Cross-validation scores: [0.96698113 0.97638725 0.98229044 0.97874852 0.95395514]
Mean cross-validation score: 0.9716724956004544
```

Figure 13: Cross-validation score

(Source: Self-Created)

To check the model generalization, K-fold cross-validation was used with K=5. The cross-validation scores of estimators were high for the most part with a mean score of 0. 9717. This is an indication that the model is reliable and firm with various folds of the data set in this case different folds of data.

d) Model Selection

Thus, the selection of the best model was conducted based on the results of hyperparameter tuning and cross-validation, and it was the Random Forest model. The capacity of the model to deal with large sets of data, coupled with the high precision and performance, tends to make it comprehensive for solving the business issue in question, that is loan approval prediction (Wu *et al.* 2023).

Task 4: Business Decision and Recommendation

The analysis and model predictions provide several actionable insights for the business:

Focus on High CIBIL Scores: Loans that are granted are those that are applied with higher CIBIL scores since applicants with high scores have higher chances of having their loans approved. Stressing on the above criterion in the assessment can also minimise impediments to approvals as well as chances of default.

Asset Value Consideration: They include the evaluation of prospective home, commercial use for business, and luxury asset approval which determines the value of the loan. It is imperative to argue that including such values in the decision-making framework can improve the evaluation of the applicant’s financial solvency (Zhang *et al.* 2024).

Model-Driven Decisions: Using the Random Forest model the loan approval process can be smoothed out require less human interaction and be faster. Applying this model as a part of the staff work in loan evaluation would enhance the performance and supplied better results.

Thus, the Random Forest model applied by following a careful procedure including data preprocessing, hyperparameter tuning, and performance evaluation can be considered an efficient approach for loan approval status prediction (Zinovieva *et al.* 2023). The use of the system in business processes seeks to increase the ability to provide accurate decisions to support operations.

Conclusion

The detailed process of creating and testing a classification model has been refined through this process, and the Random Forest classifier is the best model for predicting the loan approval status. Through accuracy assessment as well as the solid performance measures it produces, the model is capable of facilitating the automation of the loan approval process. Many tips like CIBIL score importance and asset value have been pointed out to help make improvements in the loan valuation. At the same time, the application of this model can lead to the simplification of work, the exclusion of many manual errors, and, therefore, the improved assessment of credit proposals, which will positively affect the decision-making in the field of financing and the management of credit risks.