**Loan Application Status Analysis**

**Submitted by:**

**Ruchita Singh**

**Acknowledgment**

Throughout the completion of this project, I consulted various sources including GitHub and reference materials from Data Trained institute.

**Introduction**

**Problem Statement:**

This dataset includes details of applicants who have applied for loan. The dataset includes details like credit history, loan amount, their income, dependents etc.

Independent Variables:

1. Loan\_ID - This refer to the unique identifier of the applicant's affirmed purchases
2. Gender - This refers to either of the two main categories (male and female) into which applicants are divided on the basis of their reproductive functions
3. Married - This refers to applicant being in a state of matrimony
4. Dependents - This refers to persons who depends on the applicants for survival
5. Education - This refers to number of years in which applicant received systematic instruction, especially at a school or university
6. Self\_employed - This refers to applicant working for oneself as a freelancer or the owner of a business rather than for an employer
7. Applicant Income - This refers to disposable income available for the applicant's use under State law.
8. CoapplicantIncome - This refers to disposable income available for the people that participate in the loan application process alongside the main applicant use under State law.
9. Loan\_Amount - This refers to the amount of money an applicant owe at any given time.
10. Loan\_Amount\_Term - This refers to the duration in which the loan is availed to the applicant
11. Credit History - This refers to a record of applicant's ability to repay debts and demonstrated responsibility in repaying them.
12. Property\_Area - This refers to the total area within the boundaries of the property as set out in Schedule.
13. Loan\_Status - This refers to whether applicant is eligible to be availed the Loan requested.

You have to build a model that can predict whether the loan of the applicant will be approved(Loan Status) or not on the basis of the details provided in the dataset.

**Analytical Problem Framing**

Mathematical/Analytical Modeling of the Problem:-

Throughout the project development process, we conducted thorough statistical analysis on all available attributes and meticulously examined the data structure. The following tasks were undertaken from a data-centric perspective:

1. Comprehensive analysis of data types to understand the nature and characteristics of the available data.

2. Utilization of visual data analysis techniques to gain insights into patterns, trends, and distributions within the dataset.

3. Conducting correlation analysis to identify relationships and dependencies between different variables, aiding in feature selection and model building.

4. Employing outlier detection methods to identify and handle anomalous data points that could potentially impact the model's performance.

5. Analysis and definition of the 'target' variable, which serves as the focal point for predictive modeling and decision-making.

Based on the insights and findings garnered from these steps, we gained a deeper understanding of the variables that will be generated during the data preparation stage and formulated a clearer vision for the system architecture. This informed approach ensures that our subsequent modeling efforts are grounded in a robust understanding of the data and its underlying structure.

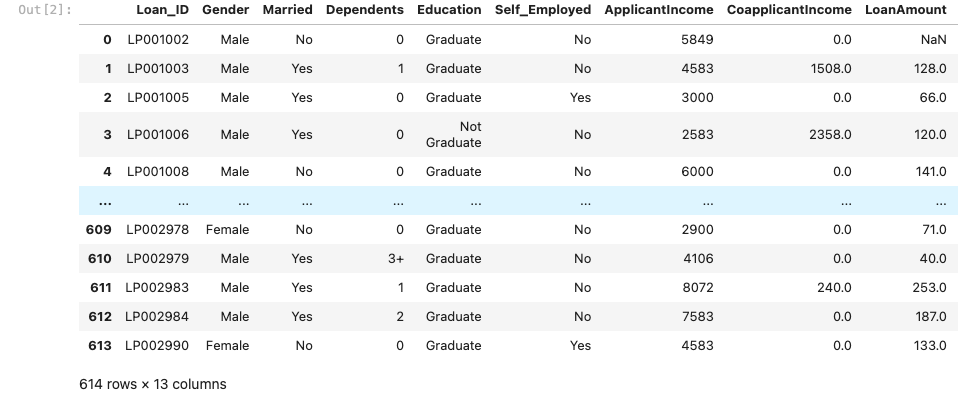
**Data Analysis -**

The dataset to be utilized for our analysis can be accessed via the following link:

<https://github.com/dsrscientist/DSData/blob/master/loan_prediction.csv>

The objective of this project is to develop a predictive model that determines whether a loan applicant's loan will be approved or not based on the information provided in the dataset.

The dataset comprises 13 independent variables, spanning from Loan\_ID to Loan Status.



The dataset consists of a total of 614 rows and 13 columns.



The data types of the variables in the dataset are as follows:

* Loan\_ID, Gender, Married, Dependents, Education, Self Employed, Property Area, Loan Status: Object
* Applicant Income: Integer
* -Co-applicant Income, Loan Amount, Loan Amount Term, Credit History: Float
* You can use ‘df.dtypes’ or ‘df.info()’ to obtain information about the data types of each column in the Data Frame.

A screenshot of a computer

Description automatically generated

In the provided dataset for loan applications, the variables can be categorized as either numeric or categorical:

Numeric Data:

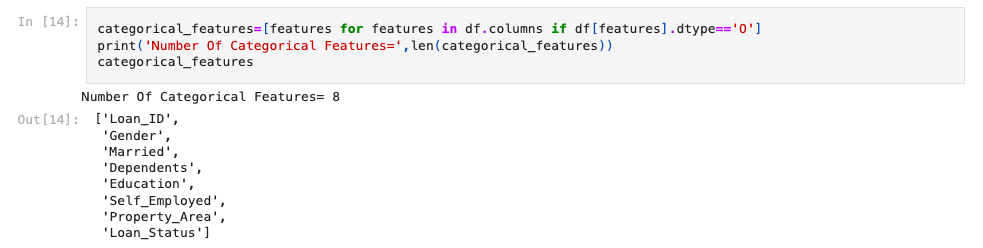
* ApplicantIncome
* CoapplicantIncome
* LoanAmount
* Loan\_Amount\_Term
* Credit\_History

A screen shot of a computer

Description automatically generated

Categorical Data:

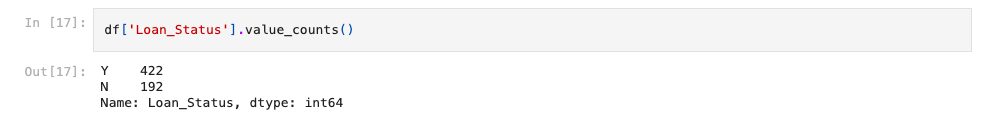
* Loan\_ID
* Gender
* Married
* Dependents
* Education
* Self\_Employed
* Property\_Area
* Loan\_Status



Understanding this differentiation is crucial for pre-processing and analyzing the data effectively.

**Visualization and Data Inputs-Logic-Output Relationship**

Understanding the relationship between input parameters (features) and output (labels/target values) is crucial. In this project, 'Loan\_Status' depends on input features like Gender, Marital status, Dependents, etc. Visualizing data helps identify patterns. Analyzing feature logic aids in predicting correct target values. Ultimately, the goal is to build a model accurately predicting loan approval based on input features.



A screenshot of a computer

Description automatically generated

A screen shot of a graph

Description automatically generated

A screen shot of a graph

Description automatically generated

A screen shot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

We'll visualize categorical variables using countplots to understand their distribution and frequency within the dataset.

In the provided loan application dataset, I'm applying pairplot graphs to analyze the relationships between different columns. This helps in visualizing the pairwise relationships between numerical variables and provides insights into potential correlations or patterns in the data.

import seaborn as sns

import matplotlib.pyplot as plt

sns.pairplot(data=df, color='blue')

plt.show()

A screenshot of a computer screen

Description automatically generated

The `df.describe()` method provides statistical details such as count, mean, standard deviation, minimum, maximum, and percentiles (25%, 50%, 75%) for the DataFrame.

This method offers a quick overview of the numerical variables' distribution and central tendency measures.

A screenshot of a computer screen

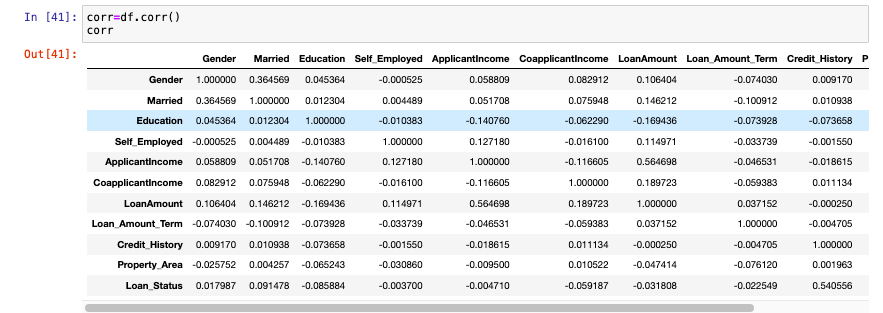
Description automatically generated

The outcome indicates the presence of outliers in the dataset, as evidenced by the significant difference between the mean and the median (50%) values. This disparity is noticeable in variables such as Married, Dependents, ApplicantIncome, CoapplicantIncome, and LoanAmount.

**Let's visualize the summary statistics of the selected variables to identify outliers and understand their distribution better.**

****

To check the correlation of features with the label value, we utilized the `corr` method. The resulting outcome is as follows:



**Let's visualize the correlation between features and the label value:**

**A screenshot of a computer

Description automatically generated**

All columns in the database exhibit positive correlations with the target value. Specifically:

* Gender: 1.8%
* Married: 9.1%
* Dependents: 1%
* Education: -8.6%
* Self\_Employed: -0.37%
* ApplicantIncome: -0.47%
* CoapplicantIncome: -5.9%
* LoanAmount: -4.1%
* Loan\_Amount\_Term: -0.57%
* Credit\_History: 54%
* Property\_Area: 3.2%

The maximum correlation is observed with Credit\_History, while the minimum correlation is with Education.

**Data Pre-processing**

During this stage, the primary objective is to prepare the data for machine learning modeling. This involves aggregating data properly and creating all available variables. Additionally, it's crucial to define the target variable.

In the data processing stage, we conducted the following checks:

* Dimension of data: `df.shape`
* Type of data: `df.info()`
* Presence of null values: `df.isnull().sum()`

If null values were present in the dataset, they were filled using methods such as mean, median, or mode.

Visualizing null values with a heatmap using `sns.heatmap(df.isnull())` aids in better understanding the distribution of missing values within the dataset.

Null values are present in this dataset, which can be verified using `df.isnull().sum()`. Specifically, the following columns contain null values:

* Gender
* Married
* Dependents
* Self\_Employed
* LoanAmount
* Loan\_Amount\_Term
* Credit\_History

A white rectangular object with a white border

Description automatically generated with medium confidence

The outcome below illustrates the distribution of null values within the dataset. The presence of null values is indicated by the non-uniform distribution of black color.

A screenshot of a computer

Description automatically generated

In the given dataset, after applying the `fillna()` method using SimpleImputer, all null values were filled using the mean and mode methods.

A screenshot of a computer code

Description automatically generated

After filling null values with the mode method, let's recheck and visualize the dataset to ensure completeness.

A screenshot of a computer

Description automatically generated

The uniform distribution of red color indicates the absence of null values in the dataset after filling them using the mode method

A screenshot of a computer

Description automatically generated

**Label Encoding**

For string/object type of data, it is crucial to convert them into integer datatype. To achieve this, we utilize the LabelEncoder() method. In this project, the following datasets were converted from string to integer datatype: Married, Dependents, Self\_Employed, LoanAmount, Loan\_Amount\_Term, and Credit\_history.

The dataset has been successfully converted from string to integer datatype using LabelEncoder().

A screenshot of a computer

Description automatically generated

**Let's visualize the presence of outliers in the dataset using box plots**

A screenshot of a computer

Description automatically generated

You can use `df.skew()` to check for skewness present in the dataset.

A white rectangular object with a white border

Description automatically generated

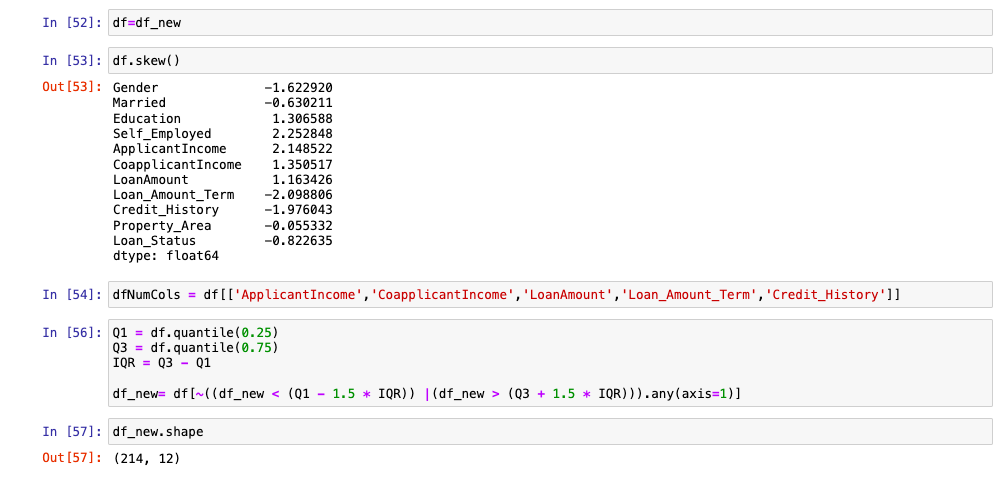
The normalized data range indicates skewness between +0.5 to -0.5. The following columns exhibit skewness:

* Gender
* Married
* Dependents
* Education
* Self\_Employed
* ApplicantIncome
* CoapplicantIncome
* LoanAmount
* Loan\_Amount\_Term
* Credit\_History
* Loan\_Status

Let's visualize the skewness using distplot.



After removing skewness using zscore, 2.44% of the data was lost.



After removing skewness using the quantile method, the new shape of the dataset is 214 rows and 12 columns, resulting in a loss percentage of 2.44%.

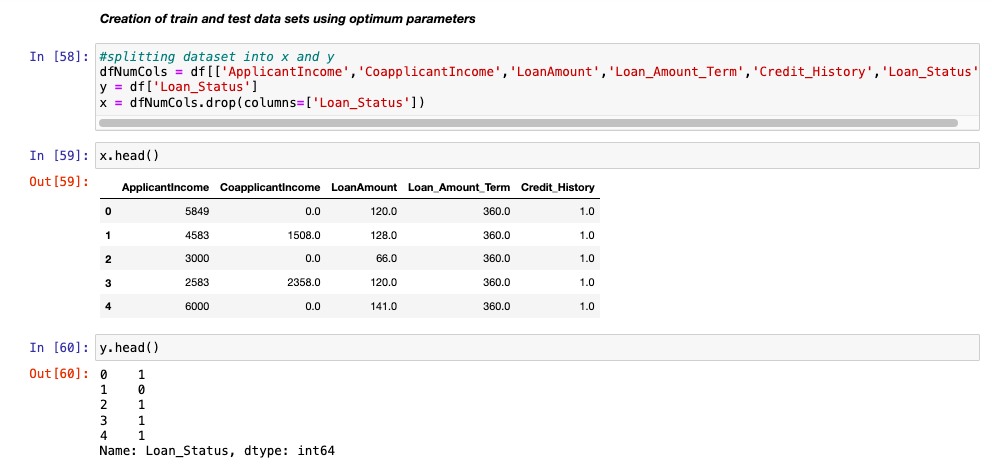
After applying the z-score method to the numerical data in the given dataset, the new shape will be 577 rows and 12 columns.

A screenshot of a computer

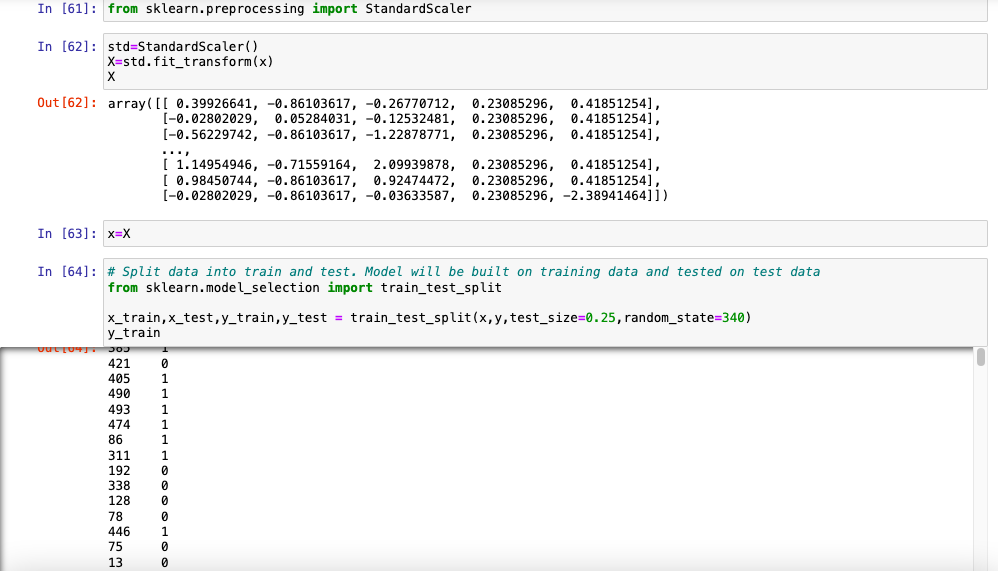
Description automatically generated

After applying the z-score method, the new dataset was obtained with a loss percentage of 6.026%.

Dividing the dataset into features and labels involves splitting it into input features (x) and the target variable (y). When considering only numeric datasets, we'll split the dataset into x and y accordingly.



Performing data standardization involves scaling the numerical features to have a mean of 0 and a standard deviation of 1. After standardization, the dataset is split into training and testing sets for model evaluation and validation purposes.



**Hardware and software prerequisites and tools employed**

Libraries utilized during model development:

1. pandas and numpy: pandas facilitates data analysis and manipulation, while numpy offers support for multidimensional array operations.
2. Matplotlib.pyplot and seaborn: These libraries aid in data visualization, enabling the creation of various plots such as scatter plots, bar graphs, and distribution plots.
3. Warnings: Helps to manage warnings and avoid unnecessary pop-ups during model execution.
4. LabelEncoder: Used to transform categorical variables into numerical format for model compatibility.
5. Zscore: Utilized to standardize numerical data and remove skewness from the dataset.
6. Classification\_report, accuracy\_report, confusion\_matrix: Evaluation metrics providing insights into model performance, including precision, recall, accuracy, and confusion matrices.
7. LogisticRegression/KNeighborsClassifier/RandomForestClassifier/ Decision TreeClassifier: Algorithms employed for model instantiation and training, offering a range of options for classification tasks.
8. Cross\_val\_score: Provides cross-validation scores to assess model generalization and performance on unseen data.
9. GridSearchCV: Utilized for hyperparameter tuning, optimizing model performance by systematically searching through a parameter grid.
10. Pickle: Enables the saving and loading of trained models, ensuring model persistence and reproducibility.

**Model Development and Evaluation Process**

**Identification of Potential Problem-Solving Approaches (Methods)**

At this stage, it's crucial to create a robust machine learning model following best practices. This involves:

* Data pre-processing: Clean and transform data into the appropriate format.
* Feature selection: Choose the most relevant set of variables.
* Selecting appropriate metrics: Measure the performance of the model effectively.
* Training multiple models.
* Validating the stability of the model.
* Analyzing the results of the model.

**Testing of Identified Algorithms**

List of algorithms employed in the models :

1. Random State Algorithms: These algorithms include methods such as Random Forest, which utilize randomization in the model-building process to improve performance and reduce overfitting.
2. Logistic Regression: A linear model used for binary classification tasks, which estimates the probability of a certain class given input features.
3. K-Nearest Neighbors Classifier: A non-parametric method used for classification tasks, which predicts the class of a data point based on the majority class of its nearest neighbors in the feature space.
4. Random Forest Classifier: A ensemble learning method that constructs multiple decision trees during training and combines their predictions to improve accuracy and reduce overfitting.
5. Decision Tree Classifier: A non-parametric supervised learning method used for classification tasks, which predicts the target variable by learning simple decision rules inferred from the input features.

**Now that our data is prepared, we are ready to apply it to the model.**

Exploring Different Models:

1. **Random State Algorithm**:

A screenshot of a computer code

Description automatically generated

**Random State Algorithm** - The random state parameter in machine learning algorithms is crucial for generating random permutations during data splitting. By providing a specific random state value, we ensure reproducibility of results as the random number generator will produce the same sequence of random numbers. This is particularly useful for debugging and ensuring consistency in model performance across different runs. In scikit-learn, the random state parameter ensures that random numbers are generated in the same order, leading to consistent results.



In the given dataset, I implemented the Random State Algorithm using function methods to apply various machine learning algorithms.

1. **Logistic Regression** is a classification model used to predict the probability of classification outcomes. It's particularly effective for binary classification problems, where it models the probability of a certain class given input features. Unlike tree-based models, such as decision trees, its structure is linear, with internal nodes representing features of the dataset and branches representing decision rules. The outcome is represented at each leaf node.

A screenshot of a computer

Description automatically generated

1. The **Decision Tree Classifier** (DTC) is a versatile algorithm that can be applied to both classification and regression tasks, although it's primarily used for classification problems. Its structure is tree-based, with internal nodes representing features of the dataset and branches representing decision rules. At each leaf node, the outcome or prediction is represented.

A screenshot of a computer

Description automatically generated

1. The **K-Nearest Neighbors Classifier** is a classification model that predicts the class of a data point by considering the majority class among its nearest neighbors in the feature space. Specifically, it looks for the 5 nearest neighbors to the given data point and assigns the class label based on the majority class among these neighbors. This algorithm is based on the principle of similarity, where data points with similar features are likely to belong to the same class.

A screenshot of a computer

Description automatically generated

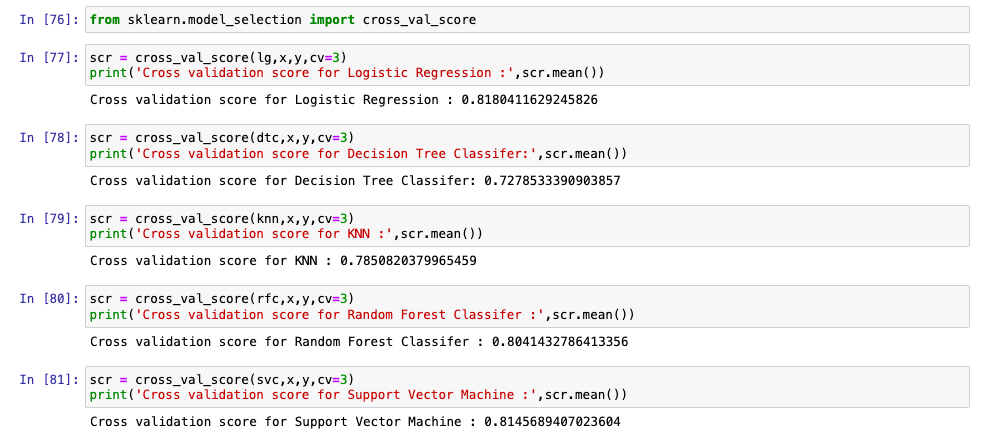
1. The **Random Forest Classifier** is an ensemble learning method for classification. It constructs multiple decision trees during training and combines their predictions to improve accuracy and reduce overfitting. By aggregating the predictions of these trees, random forests provide robust and accurate classification results.

A screenshot of a computer

Description automatically generated

**Key Performance Metrics for Success in Solving the Problem at Hand**

Cross-validation score and hyperparameter tuning were employed to mitigate any overfitting of the model. Following consideration of the cross-validation scores, the RandomForestClassifier emerged as the best model. Subsequently, hyperparameter tuning was conducted on the RandomForestClassifier, resulting in a model score of 80.60%.



**Hyp¯¯erparameter Tuning**

**Hyperparameter tuning**, also known as hyperparameter optimization, is a technique used in machine learning to find the optimal parameters of a given algorithm that yield the best performance as measured on a validation set. I employed GridSearchCV for hyperparameter tuning, which systematically searches through a grid of hyperparameters to identify the combination that produces the highest model performance.



The score improved from 80.689 to 80.7862 after hyperparameter tuning.

**Visualizations are essential for gaining insights from data and conveying findings effectively**

ROC curves are a widely-used tool in binary classification tasks, showcasing the trade-off between the true positive rate (TPR) and false positive rate (FPR). In an ROC curve, the top-left corner represents the ideal point, with a false positive rate of zero and a true positive rate of one. The steepness of the curve is crucial, as maximizing the true positive rate while minimizing the false positive rate is desirable.

While ROC curves are commonly applied in binary classification, they can also be extended to multi-label classification by binarizing the output. In this scenario, one ROC curve can be drawn per label, or alternatively, by considering each element of the label indicator matrix as a binary prediction. This approach enables the evaluation of the classifier's performance across multiple labels.

**ROC curves for the respective models in this dataset are as follows:**

**A screenshot of a computer

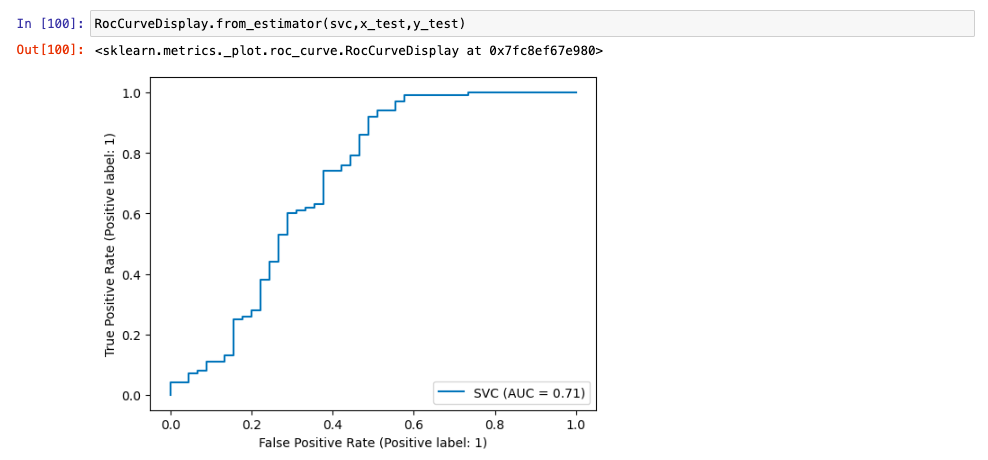
Description automatically generated**

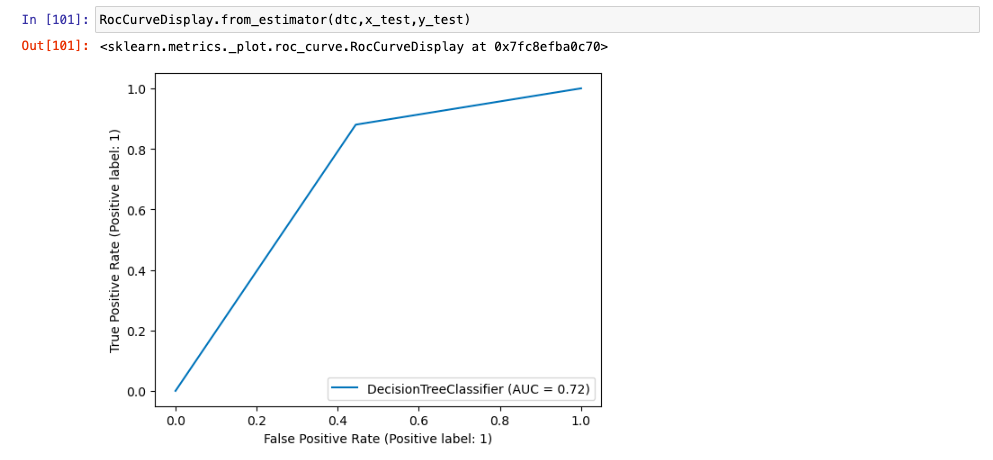
**A screen shot of a graph

Description automatically generated**

**A screen shot of a graph

Description automatically generated**

****

****

**A screen shot of a graph

Description automatically generated**

**CONCLUSION**

1. Male applicants who are married tend to have higher applicant income compared to female applicants who are married, who have the least applicant income.
2. Male applicants with a graduate degree tend to have higher applicant income compared to those without a degree.
3. Married applicants with a graduate degree have higher applicant income.
4. Non-self-employed applicants generally have higher applicant income than self-employed applicants.
5. Applicants with fewer dependents tend to have higher applicant income compared to those with more dependents.
6. Applicants owning property in urban areas and with a good credit history tend to have the highest applicant income.
7. Graduated applicants with a good credit history tend to have higher applicant income.
8. Loan amount shows a linear dependence on applicant income.
9. Applicant income and loan amount are highly positively correlated, as shown in the heatmaps.
10. The number of male applicants is higher than the number of female applicants.
11. The number of married applicants is higher than the number of unmarried applicants.
12. Applicants with no dependents are the most numerous.
13. The number of applicants with a graduate degree is higher than those without a degree.
14. Property areas are predominantly found in semi-urban areas and least in rural areas.