



DATA SCIENCE INTERNSHIP

PROJECT ON

Predicting the Likelihood of Loan Repayment for a Microfinance Institution

Presented By

Afrah Riyas

{Arunachala College of Engineering for Women}

Introduction:

MFI stands for microfinance institution, and it refers to a comprehensive package of financial services provided to the underprivileged, particularly those living in rural areas. MFIs have made MFS provision a standard practice in recent years because it is less time-consuming and expensive. However, implementing the MFS is not easy at all. Our client is a telecom business that interacts with the MFI to provide microcredit on mobile users' balances.

Problem Statement:

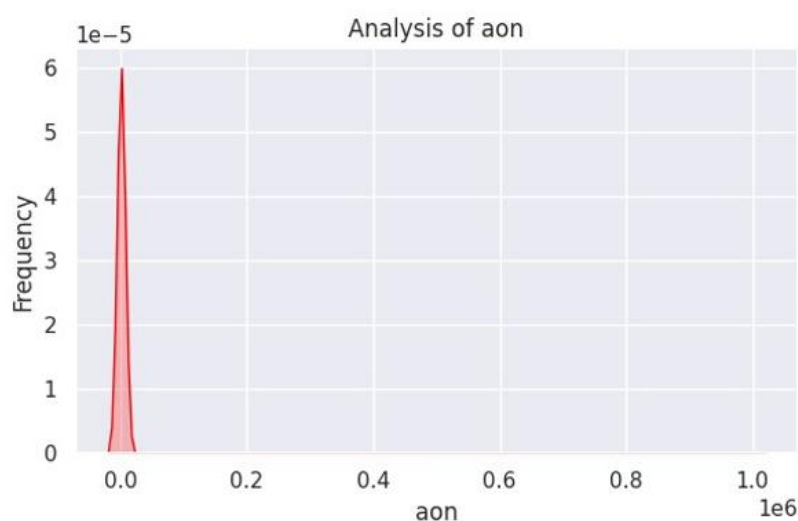
The purpose is to anticipate if clients are likely to repay their loan within 5 days. This is represented by the labels '1' for repayment and '0' for default.

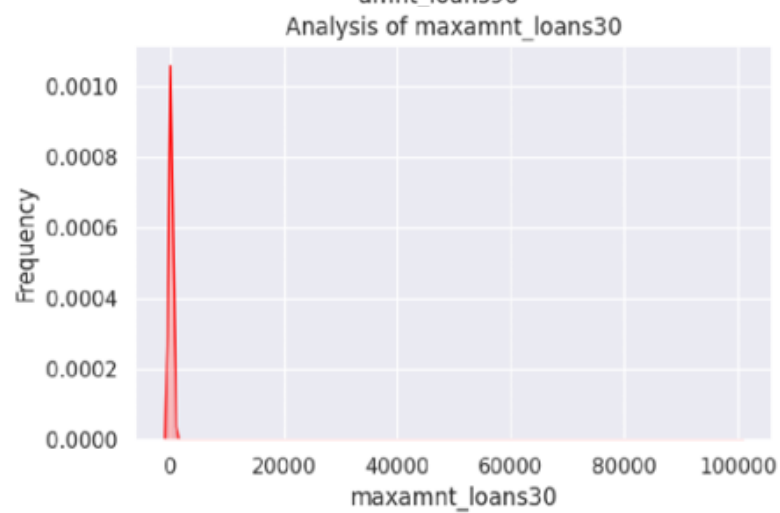
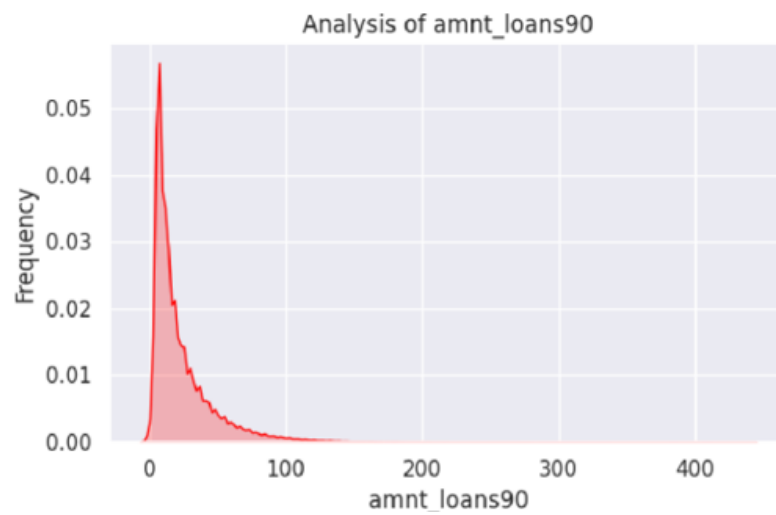
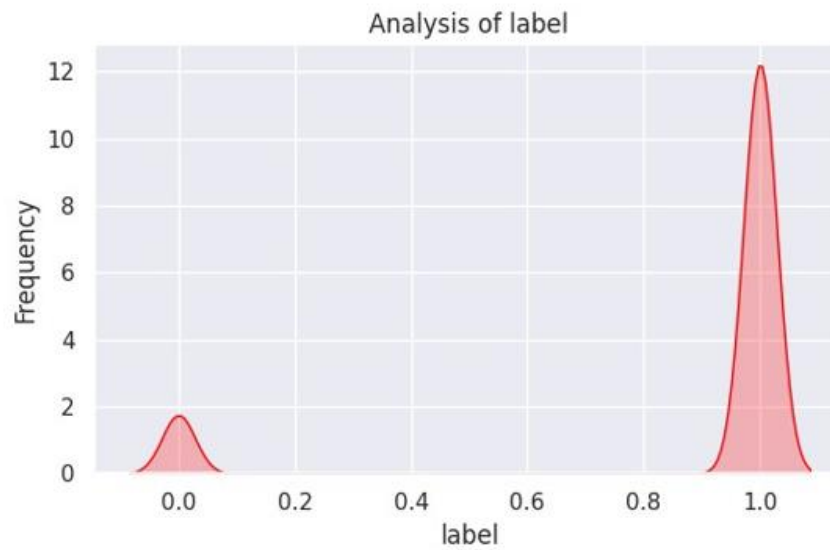
Data Preparation and Cleaning:

- Collect historical loan transaction data, including features such as 'Unnamed: 0', 'label', 'msisdn', 'aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_date_da', and 'last_rech_amt_ma'. 'cnt_ma_rech30', 'fr_ma_rech30', 'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30', 'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30', 'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans30', 'amnt_loans30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90', 'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback30', 'payback90', 'pcircle', 'pdate'.
- This dataset contains no NULL values.
- We remove the functionality "Unnamed:0".
- We remove the feature "pcircle" as it only has one unique value.

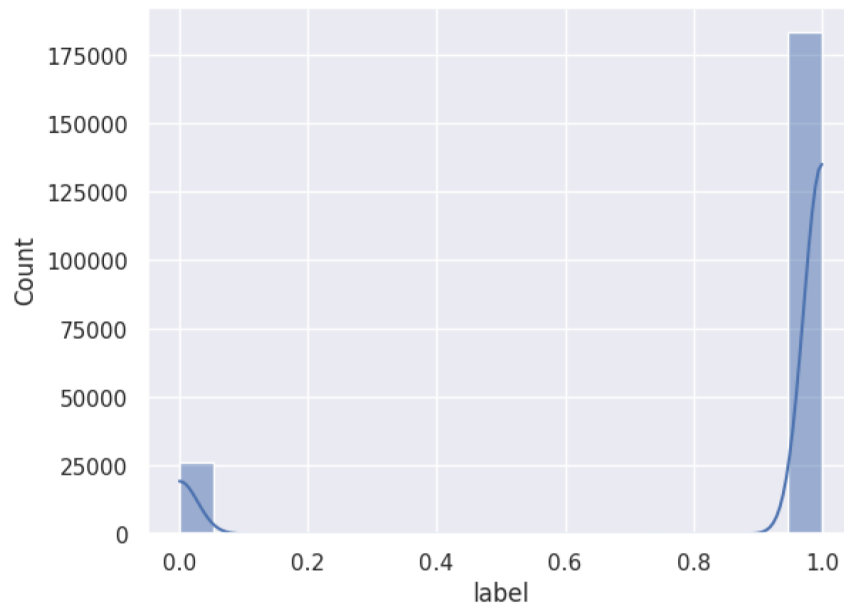
Exploratory Data Analysis:

- We visualize the univariate analysis on numerical features. Some of the Visualization are given below.





- Then we visualize the distribution of target feature.



Feature Engineering:

- We separated the temporal variable "pdate" into three new features: "Day", "Month", and "Year". & remove the "pdate" features. Convert the new features to an integer type.
- Here, the features are tied to one another. Let's discover the average of each connected feature and transform it into a single unique feature. rental30 and rental90, payback30 and payback90, cnt_ma_rech30 and cnt_ma_rech90, fr_ma_rech30 and fr_ma_rech90, sumamnt_ma_rech30 and sumamnt_ma_rech90, medianamnt_ma_rech30 and medianamnt_ma_rech90, medianmarechprebal30 and medianmarechprebal90, cnt_da_rech30 and cnt_da_rech90, fr_da_rech30 and fr
- Remove the above features.
- We encode labels for category features.
- We scale all features using Standard Scalar.
- Separate the dataset into two parts: X (all features except the target feature "label") and Y (the target feature "label").

Model Selection and Training:

- Separate X and Y into training and testing sets.
- In this classification task, we employ Logistic Regression, Decision Trees, Support Vector Machines, and Neural Networks: Multilayer Perceptron with GridSearchCV.

	Accuracy	Precision	Recall	Log Loss
Logistic Regression	0.87	0.87	1	4.57
Decision Trees	0.89	0.94	0.93	4.08
SVM	0.87	0.87	1	4.57
MLP with GridSearchCV	0.89	0.91	0.97	3.87

Output Screenshots:

```
DS-P2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Test
Logistic Regression
[ ] from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)

LogisticRegression
LogisticRegression(random_state=0)

[ ] y_pred = classifier.predict(X_test)
y_pred

array([1, 1, 1, ..., 1, 1, 1])

[ ] from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: (accuracy:.2f)")

[[ 37 2628]
 [  0 1820]]
Accuracy: 0.87

[ ] from sklearn.metrics import precision_score, recall_score, log_loss

precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
log_loss = log_loss(y_test, y_pred)

print("Precision: (precision:.2f)")
print("Recall: (recall:.2f)")
print("Log Loss: (log_loss:.2f)")

Precision: 0.87
Recall: 1.00
Log Loss: 4.57
```

```
DS-P2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Test
Decision Trees
[ ] from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(X_train, y_train)

DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)

[ ] y_pred1 = classifier.predict(X_test)
y_pred1

array([1, 1, 1, ..., 1, 1, 1])

[ ] from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred1)
print(cm)
accuracy = accuracy_score(y_test, y_pred1)
print("Accuracy: (accuracy:.2f)")

[[ 1482 1175]
 [ 1187 1730]]
Accuracy: 0.89

[ ] from sklearn.metrics import precision_score, recall_score, log_loss

precision = precision_score(y_test, y_pred1)
recall = recall_score(y_test, y_pred1)
log_loss = log_loss(y_test, y_pred1)

print("Precision: (precision:.2f)")
print("Recall: (recall:.2f)")
print("Log Loss: (log_loss:.2f)")

Precision: 0.94
Recall: 0.93
Log Loss: 4.08

Support Vector Machine
```

```
DS-P2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Test
Log Loss: 4.86
Support Vector Machine
[ ] from sklearn import svm
sv = svm.SVC(kernel='linear')
sv.fit(X_train, y_train)

SVC
SVC(kernel='linear')

[ ] y_pred2 = sv.predict(X_test)
y_pred2

array([1, 1, 1, ..., 1, 1, 1])

[ ] from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred2)
print(cm)
accuracy = accuracy_score(y_test, y_pred2)
print("Accuracy: (accuracy:.2f)")

[[  0 2657]
 [  0 1830]]
Accuracy: 0.87

[ ] from sklearn.metrics import precision_score, recall_score, log_loss

precision = precision_score(y_test, y_pred2)
recall = recall_score(y_test, y_pred2)
log_loss = log_loss(y_test, y_pred2)

print("Precision: (precision:.2f)")
print("Recall: (recall:.2f)")
print("Log Loss: (log_loss:.2f)")

Precision: 0.87
Recall: 1.00
Log Loss: 4.57
```

Model Evaluation:

	Accuracy	Precision	Recall	Log Loss
Logistic Regression	0.87	0.87	1	4.57
Decision Trees	0.89	0.94	0.93	4.08
SVM	0.87	0.87	1	4.57
MLP with GridSearchCV	0.89	0.91	0.97	3.87

Hyperparameter Tuning:

- We use GridSearchCV to tune hyperparameters in MLP Neural Network.
- We set the 'batch size' to [16, 32] and the 'learning rate' to [0.001, 0.01].

Feature importance Analysis:

- After feature engineering, use all 21 features with `"from sklearn.linear_model import LogisticRegression"` and `"from sklearn.feature_selection import SelectFromModel"`.
- Logistic regression yields coefficients for each feature's impact on the target variable (property price in this case).
- Positive coefficients imply a positive impact on price, whereas negative coefficients indicate a negative influence.
- Regularisation options for logistic regression include L1 and L2.
- Regularisation reduces overfitting and enhances feature selection.

Business Implication:

- Surprise Housing uses a predictive model to estimate property prices.
- Make investment decisions by identifying undervalued properties.
- Prioritise features, such as schools and location, to reduce risk.
- Adjust approach based on market timing and coefficient changes.
- Determine competitive prices for selling or renting.
- Assess risks associated with features.
- Marketing tip: Highlight appealing qualities in postings.

Conclusion and Future Steps:

- Surprise Housing can utilise the prediction model to make informed investment decisions, find undervalued homes, and optimise pricing tactics.
- The model's insights increase marketing efforts and risk assessment.

Limitations Encountered:

- Data Quality: The presence of mistakes or missing values in a dataset has an impact on model performance.

Market Trends

- Improve accuracy with advanced models such as Random Forest and Gradient Boosting.
- Enhance important features, such as neighbourhood mood and property history.
- Incorporate local insights, such as impending developments and zoning changes.
- Adjust dynamic coefficients to reflect market trends over time.
- Use user feedback to continuously enhance the model.

The code with output for the above Project is given below

[DS-P2.ipynb](#)