2021

LOAN INTEREST RATE PREDICTION

PROJECT DOCUMENTATION

MOUMITA MURMU

PROBLEM STATEMENT

To predict Interest Rate for customers based on the customer data provided.

o **ID:**	Unique Loan ID
o **Amount.Requested**:	Amount of money requested by the borrower
o **Loan. Length**:	Number of payments (36 or 60)
o **Loan. Purpose**:	Reason for loan provided by borrower
o **Debt.To.Income.Ratio**:	All your monthly debt payments divided by your gross monthly income
o **Home. Ownership**:	Home ownership status: (RENT, OWN, MORTGAGE, OTHERS)
o **Monthly. Income**:	Monthly income of borrower
o **Open. CREDIT. Lines**:	The number of open credit lines
o **Revolving.CREDIT. Balance:**	Type of credit Balance that can be used repeatedly up to a certain limit as long as the account is open, and payments are made on time. With revolving credit, the amount of available credit
o **Inquiries.in.the.Last.6. Months**:	A credit inquiry is a request by an institution for credit report information from a credit reporting agency.
o **Employment.Length**:	Employment period of the customer.
o **FICO**:	FICO credit score of the borrower
o **Interest. Rate** (Target Variable):	Interest Rate on the loan

EXPLORATORY DATA ANALYSIS

```
In [8]:  1 loan_intrt_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ID                              2499 non-null   float64
 1   Amount.Requested                2500 non-null   int64
 2   Amount.Funded.By.Investors      2500 non-null   float64
 3   Interest.Rate                   2500 non-null   float64
 4   Loan.Length                     2499 non-null   object
 5   Loan.Purpose                    2499 non-null   object
 6   Debt.To.Income.Ratio            2500 non-null   float64
 7   Home.Ownership                  2499 non-null   object
 8   Monthly.Income                  2500 non-null   float64
 9   Open.CREDIT.Lines               2500 non-null   int64
 10  Revolving.CREDIT.Balance        2500 non-null   int64
 11  Inquiries.in.the.Last.6.Months  2500 non-null   int64
 12  Employment.Length               2422 non-null   object
 13  fico                            2500 non-null   object
dtypes: float64(5), int64(4), object(5)
memory usage: 273.6+ KB
```

- Total Observations: 2500; Total Features = 13 (Independent) + 1 (Dependent)
- Total Features – Numerical: 9; Categorical: 5

NUMERICAL FEATURES

MEASURE OF CENTRAL TENDENCY				
Independent Feature	**Mean**	**Median**	**Min**	**Max**
Amount.Requested	12389.59	10000.00	1000.00	35000.00
Amount.Funded.By.Investors	11984.35	10000.00	200.00	35000.00
Debt.To.Income.Ratio	0.15	0.15	0.00	0.35
Monthly.Income	5685.15	5000.00	588.50	102750.00
Open.CREDIT.Lines	10.06	9.0	2.0	38.0
Revolving. CREDIT.Balance	15225.66	10938.00	0.00	270800.00
Inquiries.in.the.Last.6.Months	0.9	1.0	0.00	9.0

MEASURE OF DISPERSION & SHAPE								
Independent Feature	**Range**	**Q1**	**Q2**	**Q3**	**Variance**	**Std. Dev**	**Skewness**	**Kurtosis**
Amount.Requested	34000	6000	10000	17000	6.106517e+07	7814.42	0.9097	0.30192
Amount.Funded.By.Investors	34800	6000	10000	16000	5.996149e+07	7743.48	0.9285	0.41474
Debt.To.Income.Ratio	0.35	0.10	0.15	0.21	5.649622e-03	0.07516	0.1534	-0.5149
Monthly.Income	102161.5	3474.26	5000.00	6800.00	1.570835e+07	3963.37	8.4648	167.354
Open.CREDIT.Lines	36.0	7.00	9.00	13.00	2.040046e+01	4.51668	0.8834	1.44141
Revolving. CREDIT.Balance	270800.00	5545.25	10938.00	18870.25	3.352094e+08	18308.7	5.3795	48.8008
Inquiries.in.the.Last.6.Months	9.00	0.0	1.0	1.0	1.525382e+00	1.23506	2.0301	6.43647

Note:

- **Skewness**: (-0.5 to 0.5): Symmetrical, (-1 to -0.5 OR 0.5 to 1): Moderately Skewed, (<-1 Or >1): Highly Skewed Distribution. [Positive Skew = Positive Value = Tail is on right side of distribution, Negative Skew = Negative Values = Tail is on the left side of the distribution]
- **Kurtosis**: (k>3): Leptokurtic, (k=3): Mesokurtic, (k<3): Platykurtic Distribution.

CATEGORICAL FEATURES:

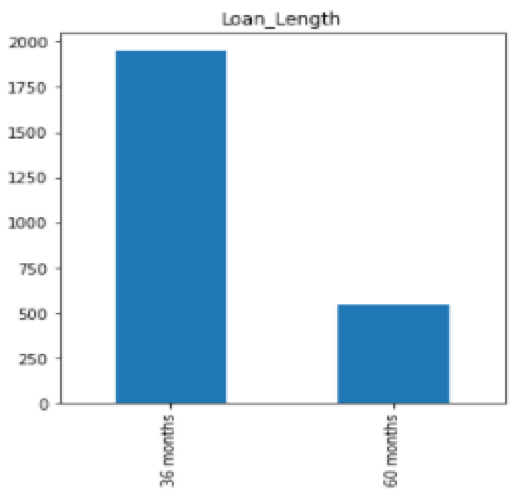
In [10]:
```
loan_intrt_data.describe(include='object')
```

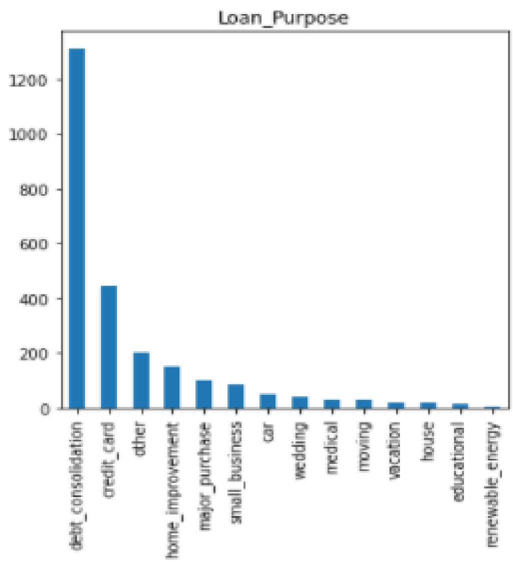
Out[10]:

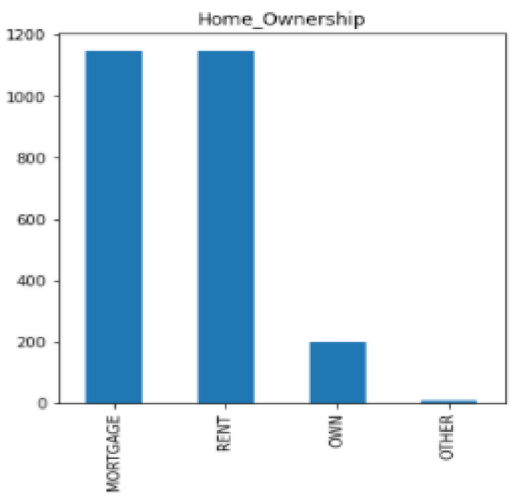
	Loan.Length	Loan.Purpose	Home.Ownership	Employment.Length	fico
count	2499	2499	2499	2422	2500
unique	3	14	5	10	38
top	36 months	debt_consolidation	MORTGAGE	10+ years	670-674
freq	1950	1307	1147	653	171

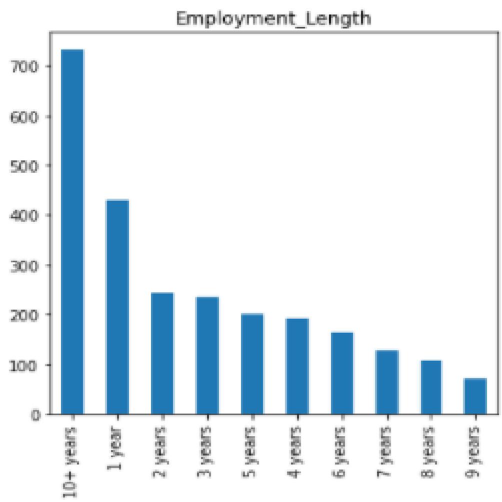
Most common value in category-

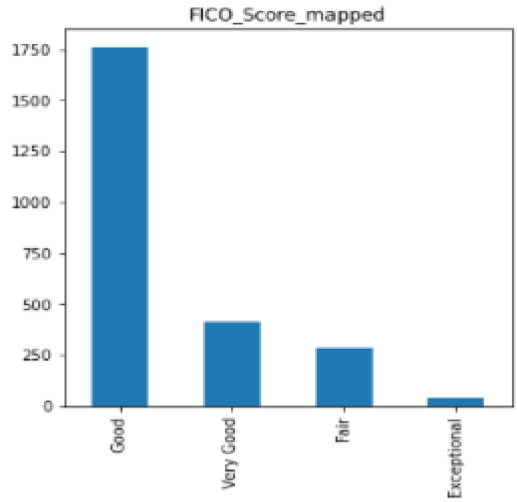
- Loan_Length: 36 Months (frequency - 1950)
- Loan_Purpose: Debt Consolidation (frequency - 1307)
- Home.Ownership: Mortgage (frequency - 1147)
- Employment_Length: 10+ Years (frequency – 653)
- FICO: 670-674 (frequency – 171)











DATA PREPROCESSING

1. MISSING VALUES IDENTIFICATION & TREATMENT:

In [17]:
```
loan_intrt_data.isnull().sum(axis=0)
```

Out[17]:
```
ID                                1
Amount_Requested                  0
Amount_Funded_By_Investors        0
Interest_Rate                     0
Loan_Length                       1
Loan_Purpose                      1
Debt_To_Income_Ratio              0
Home_Ownership                    1
Monthly_Income                    0
Open_Credit_Lines                 0
Revolving_Credit_Balance          0
Inquiries_in_the_last_6months     0
Employment_Length                78
FICO_Score                        0
dtype: int64
```

Using MEDIAN IMPUTATION (replacing missing value with median for numerical features) technique.

In [24]:
```
treat_num_columns(with_num_cols_data.columns)
```

```
Independent Variable:  Amount_Requested
Check if there are any missing value in variable Amount_Requested :
No missing values found in  Amount_Requested

Independent Variable:  Amount_Funded_By_Investors
Check if there are any missing value in variable Amount_Funded_By_Investors :
No missing values found in  Amount_Funded_By_Investors

Independent Variable:  Interest_Rate
Check if there are any missing value in variable Interest_Rate :
No missing values found in  Interest_Rate

Independent Variable:  Debt_To_Income_Ratio
Check if there are any missing value in variable Debt_To_Income_Ratio :
No missing values found in  Debt_To_Income_Ratio

Independent Variable:  Monthly_Income
Check if there are any missing value in variable Monthly_Income :
No missing values found in  Monthly_Income

Independent Variable:  Open_Credit_Lines
Check if there are any missing value in variable Open_Credit_Lines :
No missing values found in  Open_Credit_Lines

Independent Variable:  Revolving_Credit_Balance
Check if there are any missing value in variable Revolving_Credit_Balance :
No missing values found in  Revolving_Credit_Balance

Independent Variable:  Inquiries_in_the_last_6months
Check if there are any missing value in variable Inquiries_in_the_last_6months :
No missing values found in  Inquiries_in_the_last_6months
```

Using MODE IMPUTATION (replacing missing value with mode for categorical features) technique.

Checking for missing value in Categorical features and treating them

In [25]:
```
treat_cat_columns(with_cat_cols_data.columns)
```

```
Independent Variable:  Loan_Length
Check if there are any missing value in variable Loan_Length :
No. of Observations with missing value:  1
After treating the missing values
No. of Observations with missing value:  0

Independent Variable:  Loan_Purpose
Check if there are any missing value in variable Loan_Purpose :
No. of Observations with missing value:  1
After treating the missing values
No. of Observations with missing value:  0

Independent Variable:  Home_Ownership
Check if there are any missing value in variable Home_Ownership :
No. of Observations with missing value:  1
After treating the missing values
No. of Observations with missing value:  0

Independent Variable:  Employment_Length
Check if there are any missing value in variable Employment_Length :
No. of Observations with missing value:  78
After treating the missing values
No. of Observations with missing value:  0

Independent Variable:  FICO_Score
Check if there are any missing value in variable FICO_Score :
No missing values found in  FICO_Score
```

Dataset after treating the missing values-

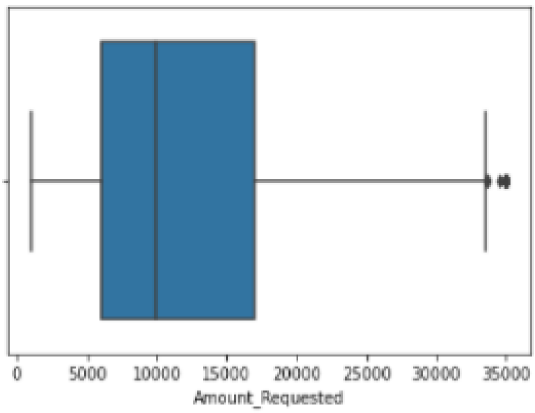
In [26]:
```
loan_data.isnull().sum(axis=0)
```

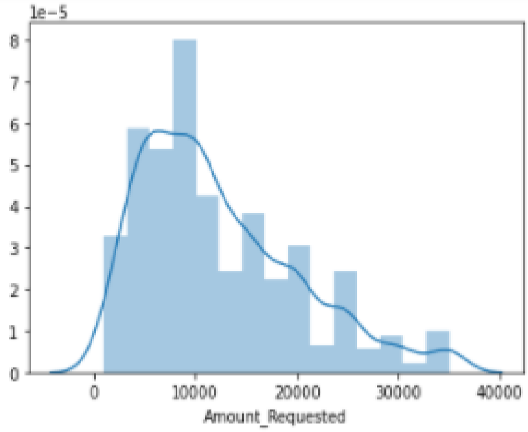
Out[26]:
```
Amount_Requested                 0
Amount_Funded_By_Investors       0
Interest_Rate                    0
Loan_Length                      0
Loan_Purpose                     0
Debt_To_Income_Ratio             0
Home_Ownership                   0
Monthly_Income                   0
Open_Credit_Lines                0
Revolving_Credit_Balance         0
Inquiries_in_the_last_6months    0
Employment_Length                0
FICO_Score                       0
dtype: int64
```

2. OUTLIERS:

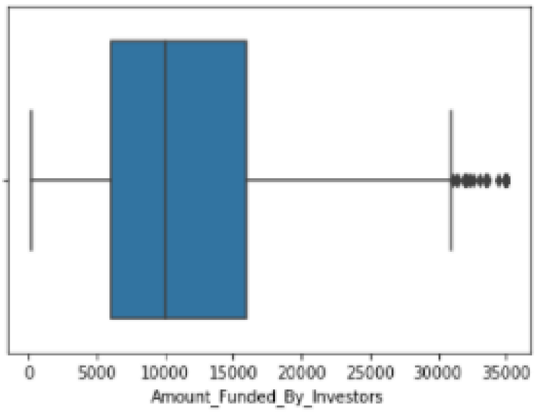
A) IDENTIFICATION

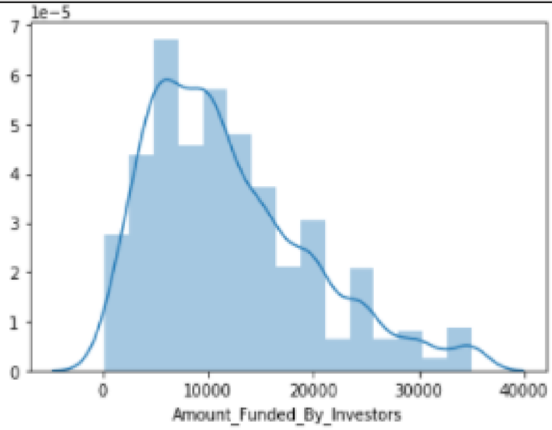
COLUMN: Amount Requested



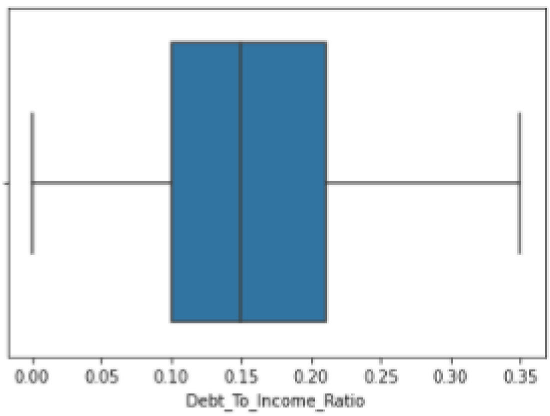


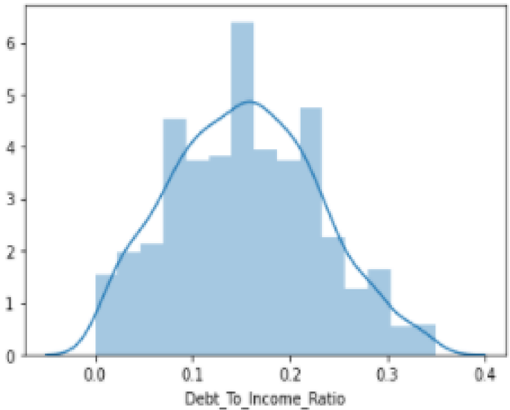
COLUMN: Amount Funded By Investors



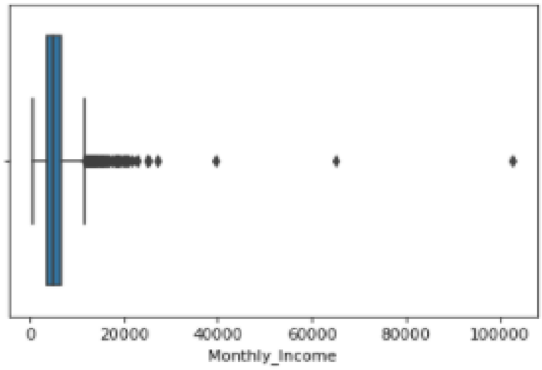


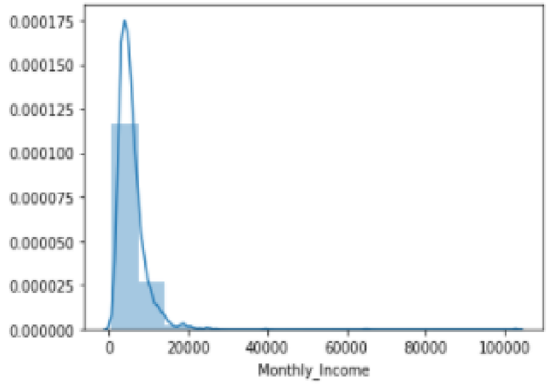
COLUMN: Debt To Income Ratio



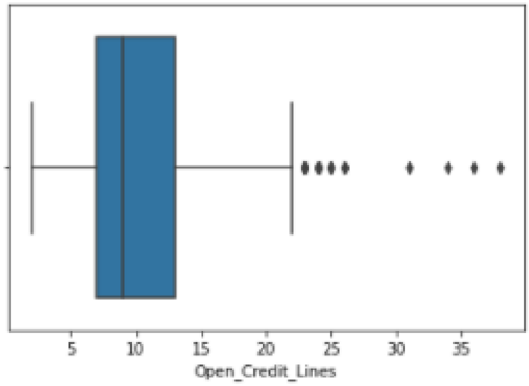


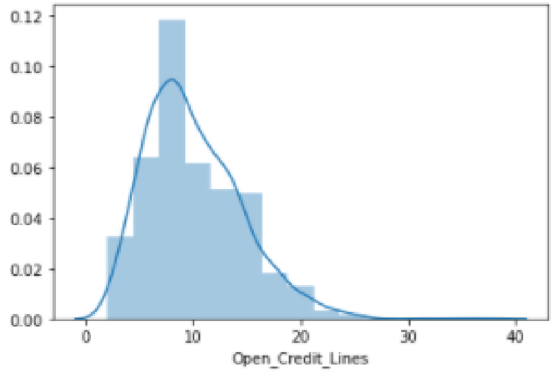
COLUMN: Monthly Income



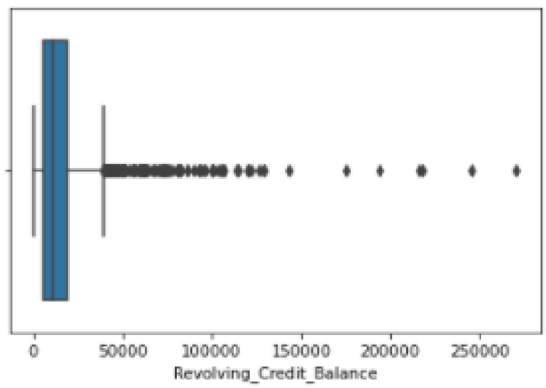


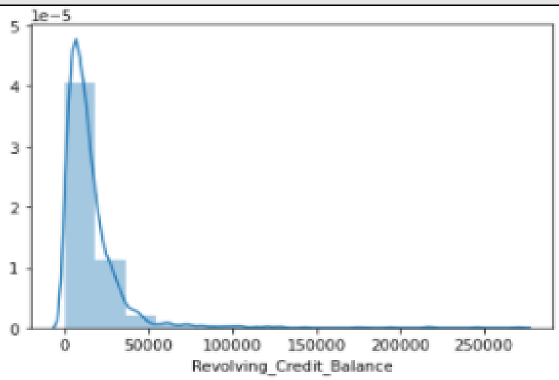
COLUMN: Open Credit Lines



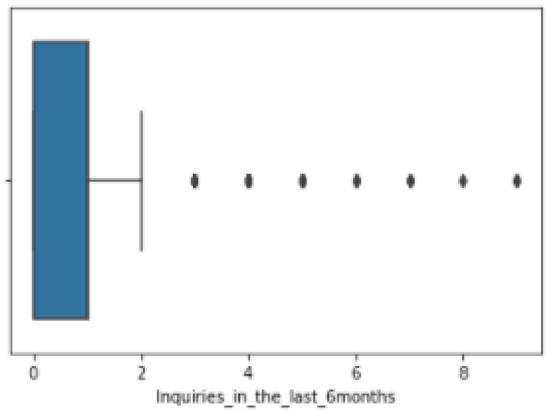


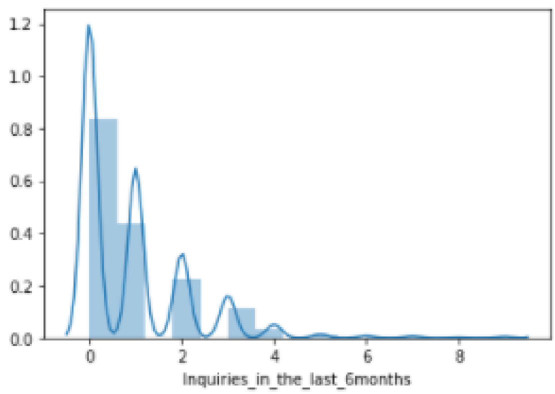
COLUMN: Revolving Credit Balance



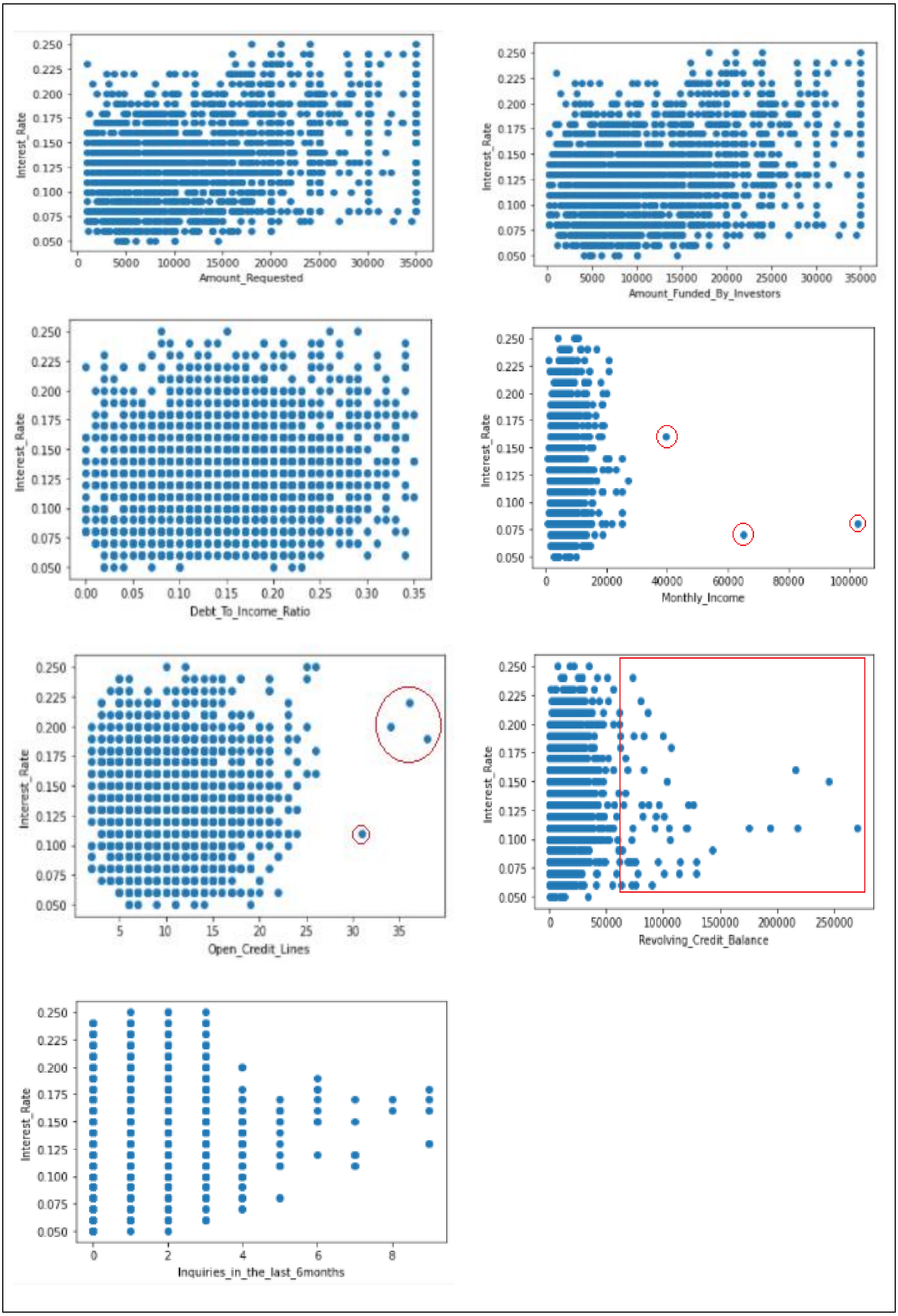


COLUMN: Inquiries in last 6 Months



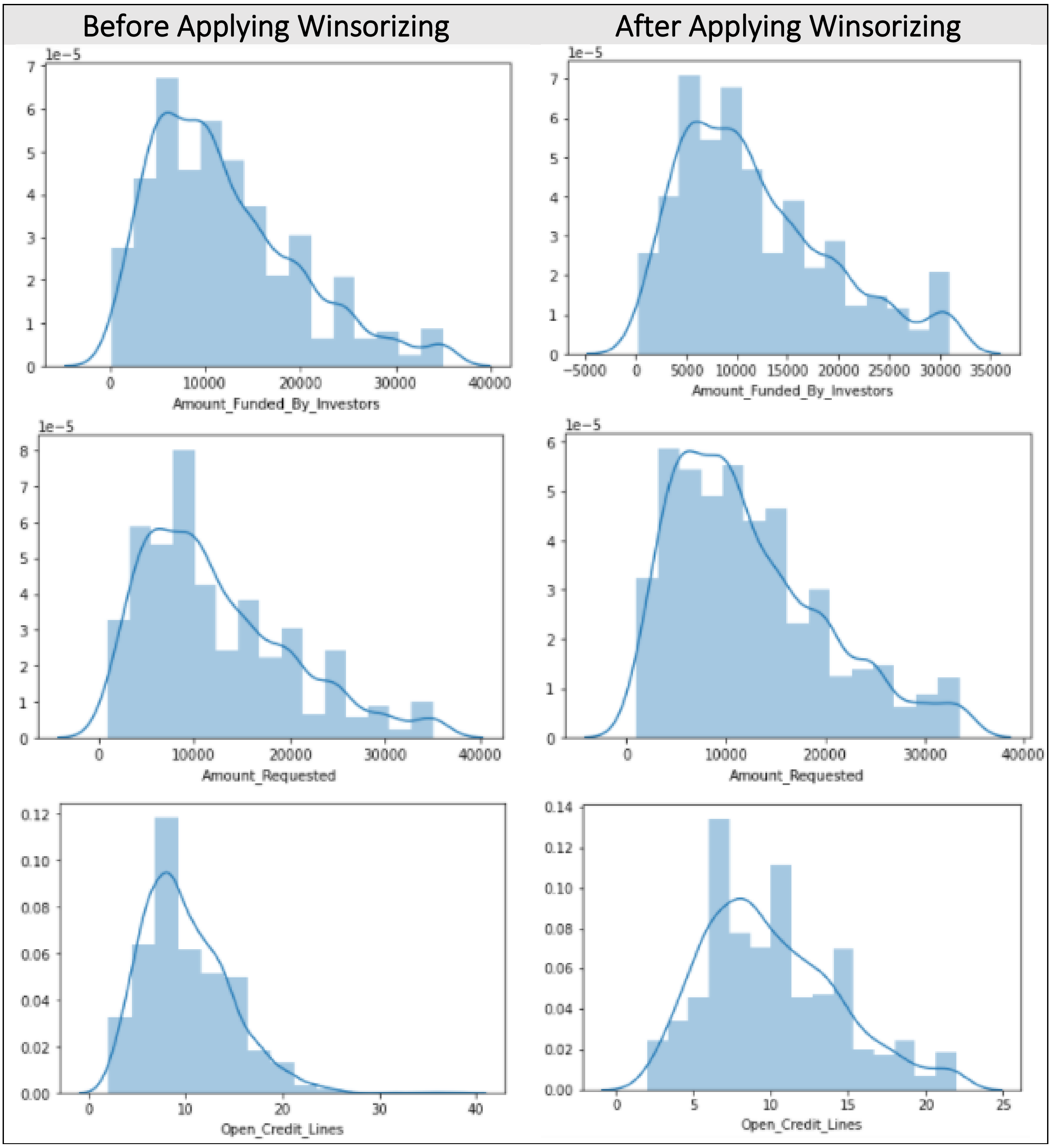


SCATTERPLOT REPRESENTATION-



B) **TREATMENT-**

Applying winsorizing techniques on features: *Amount Funded by Investors, Amount Requested, Open Credit Lines.*

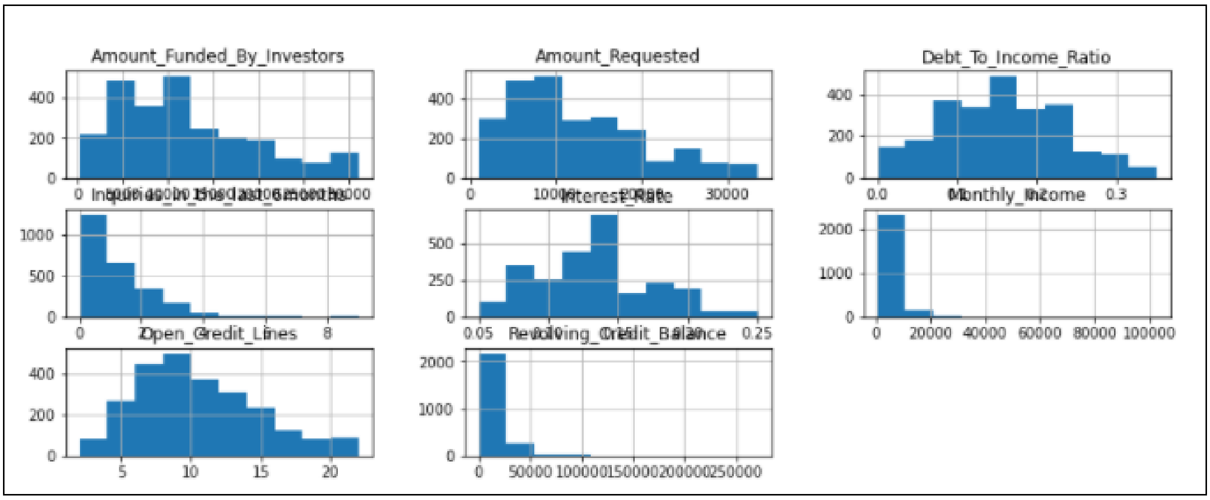


3. FEATURE ENGINEERING & SCALING:

a) VARIABLE TRANSFORMATION:

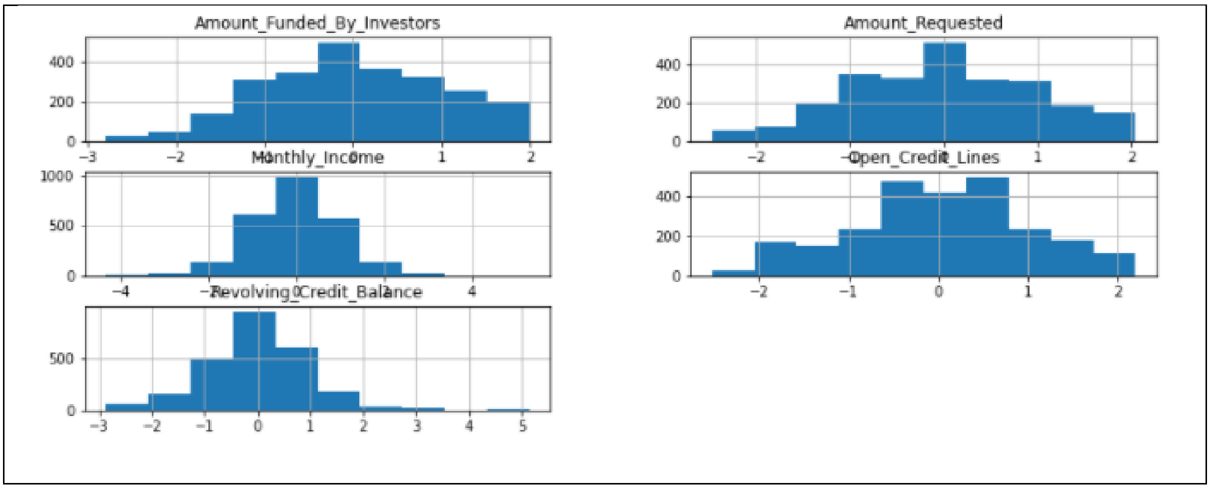
<u>FOR NUMERICAL FEATURES</u> -

Before applying feature engineering techniques on the independent variables -



– Applying POWER TRANSFORMATION on features: *Amount Funded by Investors, Amount Requested, Monthly Income, Open Credit Lines and Revolving credit balance.*

**Note: Transformation was not applied on features: Debt to income ratio & Inquiries in the last 6 months .



In [37]:
```
with_num_cols_data[cols_to_transform].var()
```

Out[37]:
```
Amount_Requested            1.0004
Amount_Funded_By_Investors  1.0004
Open_Credit_Lines           1.0004
Monthly_Income              1.0004
Revolving_Credit_Balance    1.0004
dtype: float64
```

In [38]:
```
with_num_cols_data[cols_to_transform].agg(['skew']).transpose()
```

Out[38]:

	skew
Amount_Requested	-0.040097
Amount_Funded_By_Investors	-0.038743
Open_Credit_Lines	-0.017916
Monthly_Income	-0.010300
Revolving_Credit_Balance	0.154902

FOR CATEGORICAL FEATURES –

Used ***Label Encoding*** on features: *Loan_Length, Loan_Purpose, Home_Ownership, Employment_Length.* Used ***map function*** on feature *FICO_Score* to map relevant scores.

In [49]:
```
with_cat_cols_data.head()
```

Out[49]:

	Loan_Length	Loan_Purpose	Home_Ownership	Employment_Length	FICO_Score	FICO_Score_mapped	FICO_Score_Scaled
0	0	2	0	0	735-739	Good	1
1	0	2	0	2	715-719	Good	1
2	1	2	0	2	690-694	Good	1
3	0	2	0	5	695-699	Good	1
4	0	1	4	9	695-699	Good	1

DATA PARTITION

a. Train-Test Split with 70:30 ratio
b. K-fold Cross Validation (10 Folds)

In [58]:
```
X = updated_loan_data.drop(['Interest_Rate'], axis=1)
Y = updated_loan_data['Interest_Rate']
```

In [59]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
```

In [60]:
```
X_train.shape, X_test.shape
```

Out[60]: ((1750, 12), (750, 12))

MODEL BUILDING

We will be considering the below regression algorithms and choose the model which provides better predictions for the specific regression problem statement. The models are:

a. Decision Tree Regressor
b. Random Forest Regressor
c. ADA Boost Regressor
d. Gradient Boost Regressor
e. XGBoost Regressor
f. SVR
g. KNN
h. Linear Regression

Performance Metrics (on Test Data):

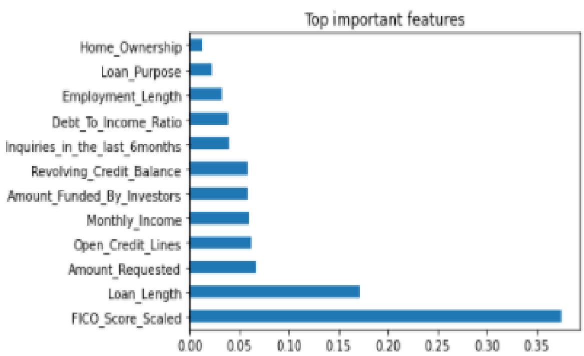
Sl No.	Performance Metrics	Decision Tree Regressor	Random Forest	Adaboost	Gradient Boosting	XGBoost	SVR	KNN	Linear Regression
1	MAE	0.0013	0.0202	0.0223	0.0196	0.0218	0.0370	0.0276	0.0210
2	MSE	0.0013	0.0007	0.0008	0.0006	0.0008	0.0020	0.0012	0.0007
3	RMSE	0.0356	0.0262	0.0277	0.0248	0.0276	0.0451	0.0340	0.0265
4	R-Squared	**0.2448**	**0.5916**	**0.5410**	**0.6320**	**0.5453**	**-0.2164**	**0.3182**	**0.5792**

Based on the scores achieved, by training the respective models, Gradient Boosting Regression model is having the highest R-squared value amongst all. Its mean squared error value and mean absolute value is lowest in comparison with the other regression models.

The important features based on Feature importance of the ML regression models are:

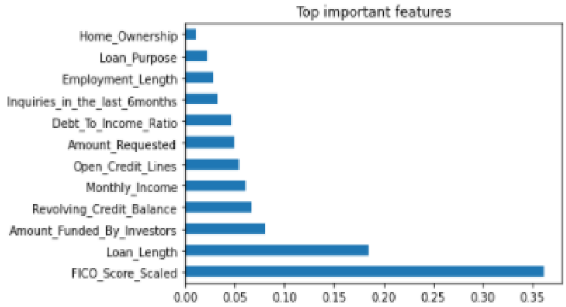
- Decision Tree Model:

```
feat_importances_dt = pd.Series(importance_dt, index=X.columns)
feat_importances_dt.nlargest(12).plot(kind='barh')
plt.title("Top important features")
plt.show()
```



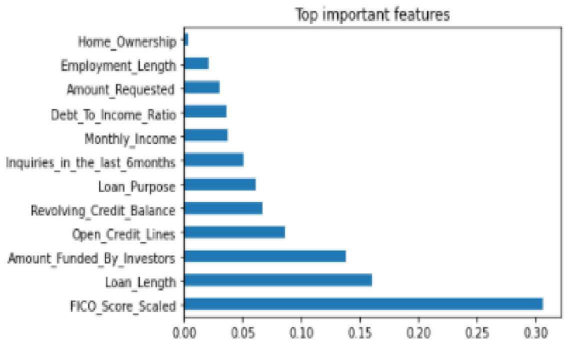
- Random Forest Model:

In [80]:
```python
feat_importances_rfreg = pd.Series(importance_rfreg, index=X.columns)
feat_importances_rfreg.nlargest(12).plot(kind='barh')
plt.title("Top important features")
plt.show()
```



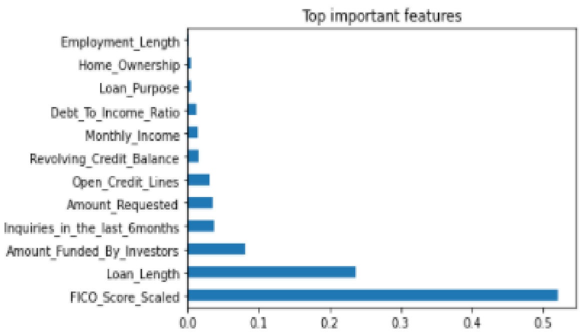
- AdaBoost Regression Model:

In [92]:
```python
feat_importances_adab = pd.Series(importance_adab, index=X.columns)
feat_importances_adab.nlargest(12).plot(kind='barh')
plt.title("Top important features")
plt.show()
```



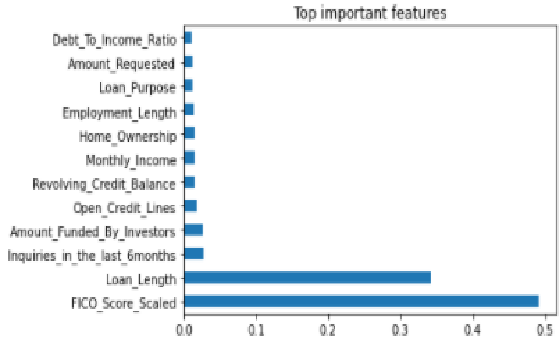
- Gradient Boosting Model

In [100]:
```python
feat_importances_gbr = pd.Series(importance_gbr, index=X.columns)
feat_importances_gbr.nlargest(12).plot(kind='barh')
plt.title("Top important features")
plt.show()
```



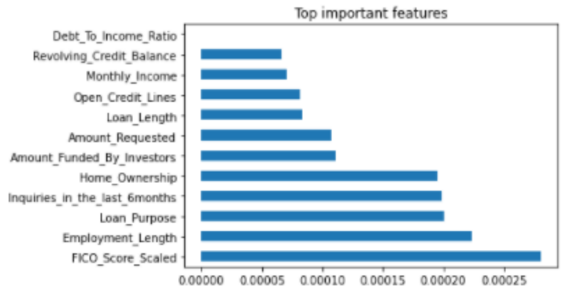
- XGBoost Model

In [108]:
```
feat_importances_xgbr = pd.Series(importance_xgbr, index=X_train.columns)
feat_importances_xgbr.nlargest(12).plot(kind='barh')
plt.title("Top important features")
plt.show()
```



- KNN Regression Model

In [116]:
```
feat_importances_knnreg = pd.Series(importance_knnreg, index=X_train.columns)
feat_importances_knnreg.nlargest(12).plot(kind='barh')
plt.title("Top important features")
plt.show()
```



- Linear Regression Model – (Backward Elimination model)

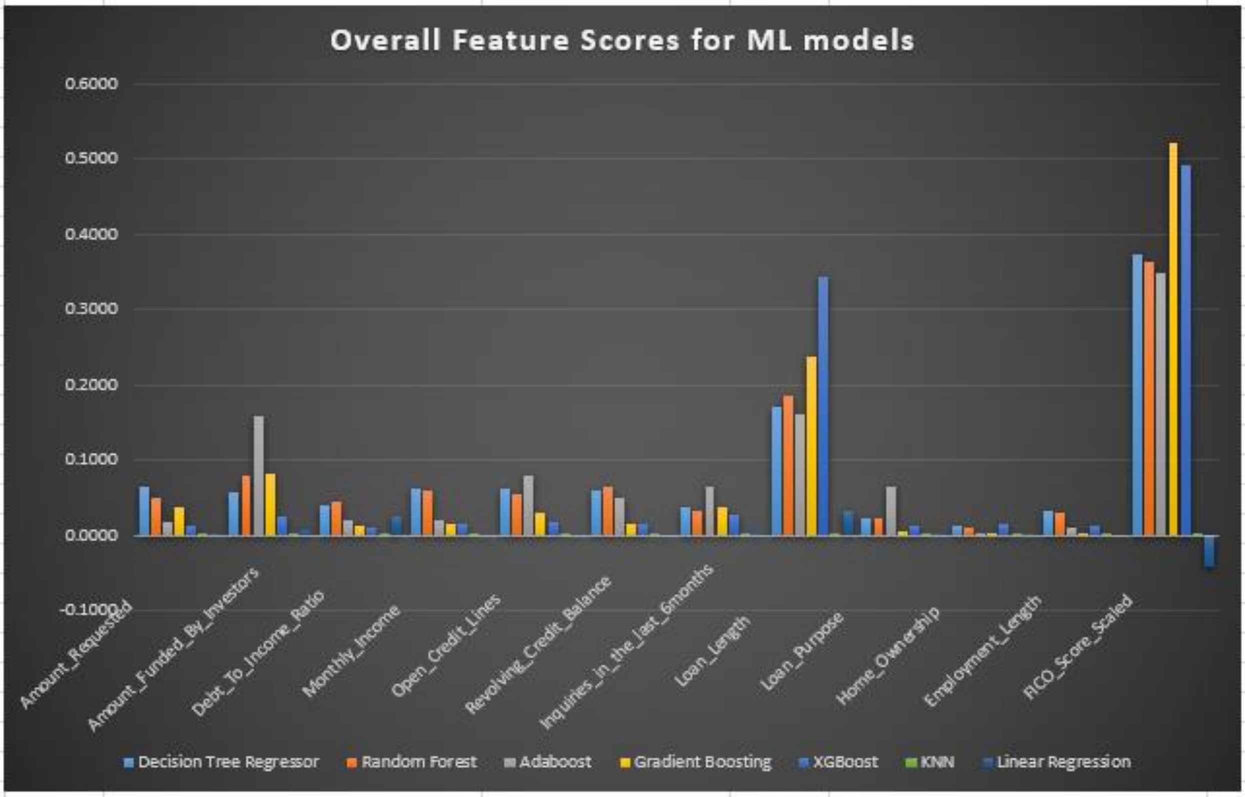
Df Residuals:	2490	BIC:	-1.103e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1565	0.002	75.382	0.000	0.152	0.161
Amount_Requested	0.0037	0.002	2.273	0.023	0.001	0.007
Amount_Funded_By_Investors	0.0070	0.002	4.368	0.000	0.004	0.010
Debt_To_Income_Ratio	0.0271	0.008	3.492	0.000	0.012	0.042
Open_Credit_Lines	-0.0025	0.001	-4.253	0.000	-0.004	-0.001
Inquiries_in_the_last_6months	0.0046	0.000	10.505	0.000	0.004	0.005
Loan_Length	0.0323	0.001	23.311	0.000	0.030	0.035
Loan_Purpose	0.0004	0.000	2.545	0.011	0.000	0.001
Home_Ownership	0.0012	0.000	4.174	0.000	0.001	0.002
FICO_Score_Scaled	-0.0419	0.001	-44.475	0.000	-0.044	-0.040

Omnibus:	21.985	Durbin-Watson:	2.006
Prob(Omnibus):	0.000	Jarque-Bera (JB):	24.155
Skew:	0.183	Prob(JB):	5.69e-06
Kurtosis:	3.313	Cond. No.	77.4

The features with p-value<0.05 are selected features.

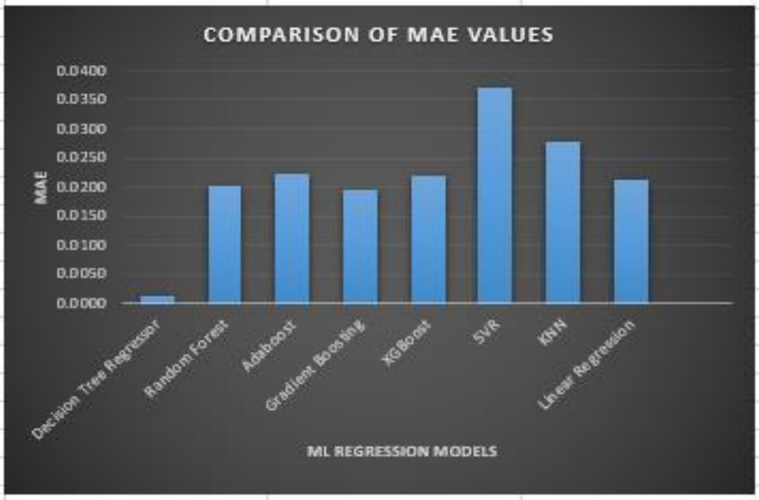
<u>To Summarize</u>:
FICO_Score_Scaled, Loan_Length, ***Amount_Funded_By_Investors*** are the top three features which are the important feature across maximum ML regression model.



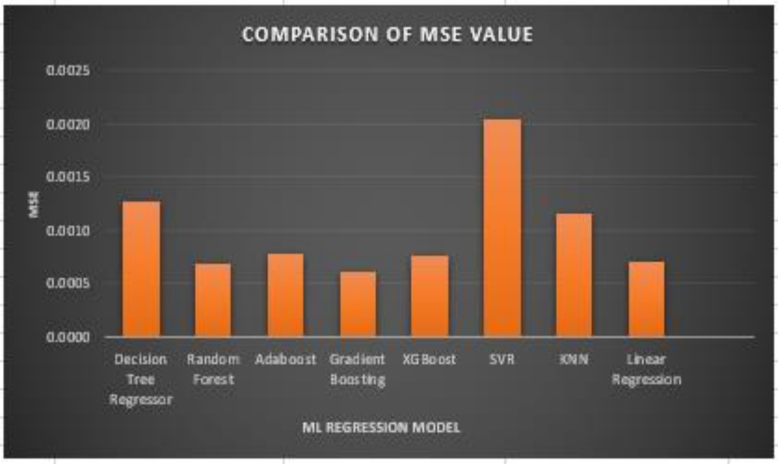
MODEL EVALUATION

Performance comparison of regression techniques without hyperparameter tuning.

- *Mean Absolute Errors of ML regression Models.*



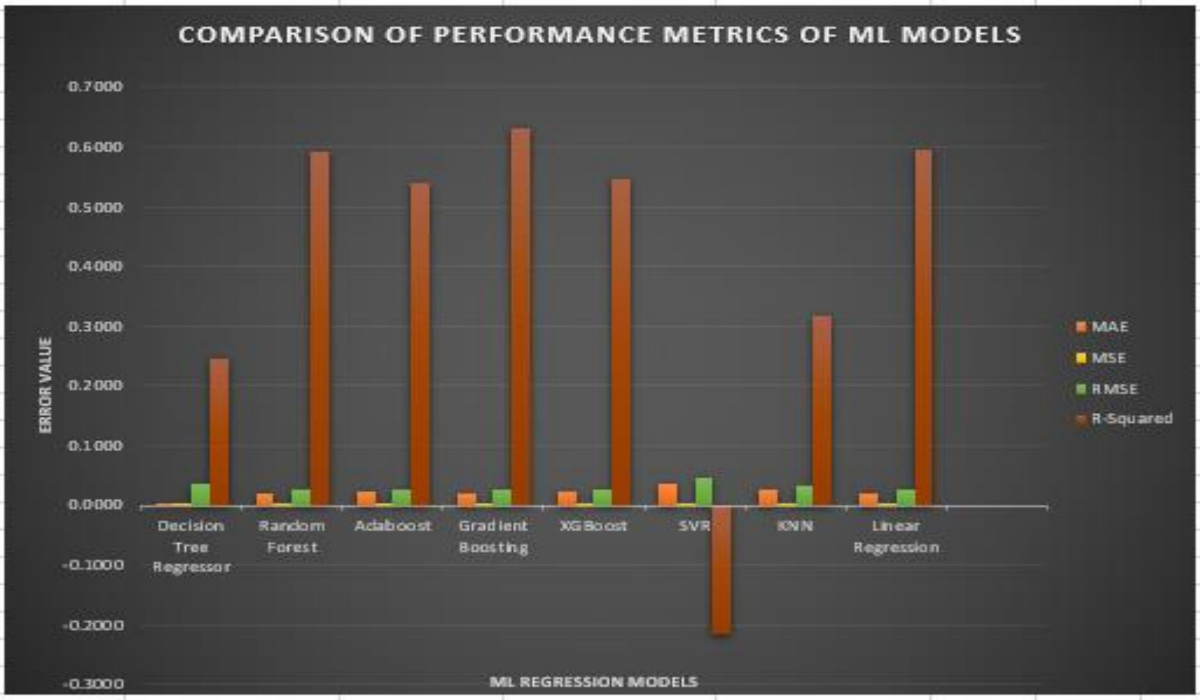
- *Mean Squared Errors of ML regression models.*



- *R-Squared Value of ML regression models*

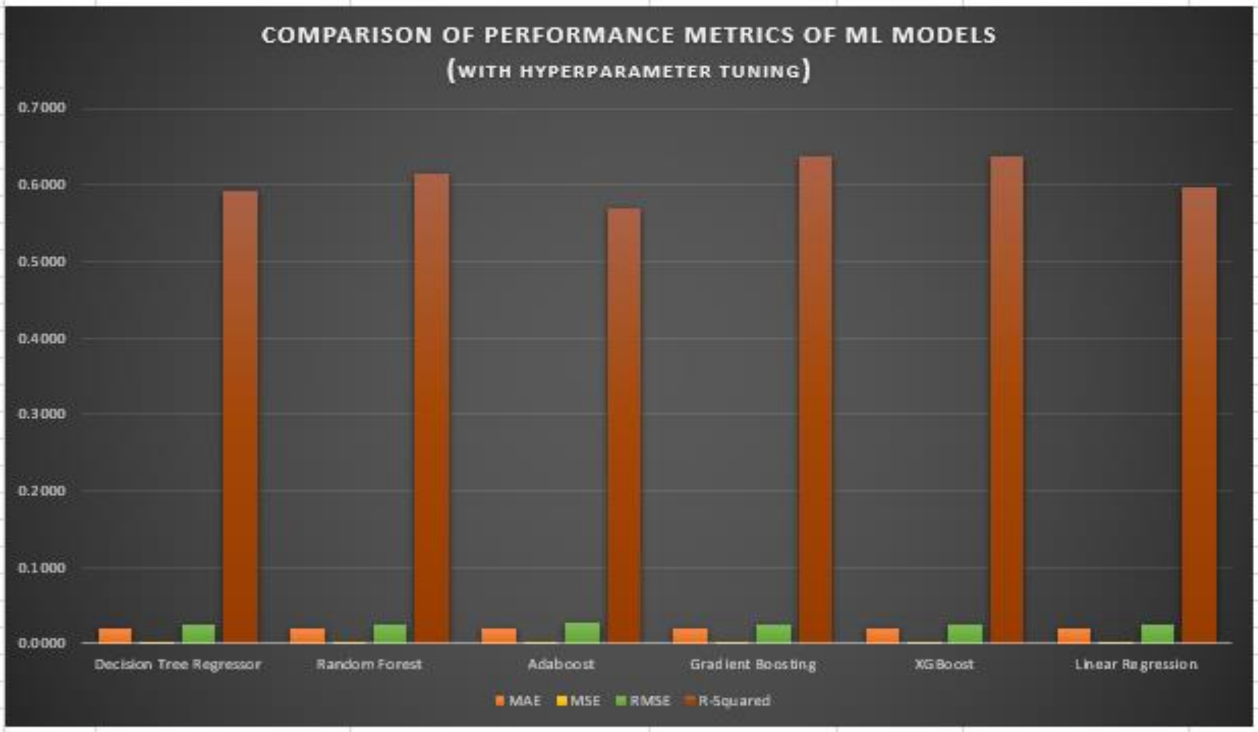
Below is the overall performance metrics (in comparison) of various ML models (without Hyperparameter tuning)-

- Out of all the Machine learning regression models considered, Gradient Boosting performed better than other models (R2 value: 0.631956376860691, MSE: 0.000616501923378396, MAE: 0.0196091910933653)
- The Mean squared error values of the Gradient Boosting models is the lowest amongst all.
- The mean absolute error value of the Gradient Boosting model is also the lowest amongst all.



Below are the performances of the ML models after applying hyperparameter tuning :

Sl No.	Performance Metrics	Decision Tree Regressor	Random Forest	Adaboost	Gradient Boosting	XGBoost	Linear Regression
1	MAE	0.0202	0.0197	0.0213	0.0196	0.0195	0.0208
2	MSE	0.0007	0.0006	0.0007	0.0006	0.0006	0.0007
3	RMSE	0.0261	0.0254	0.0268	0.0248	0.0246	0.0262
4	R-Squared	0.5924	0.6146	0.5699	0.6363	0.6374	0.5960



- The R-squared values of the considered ML models, after the hyperparameter tuning, have increased.
- The results showed a decrease in the prediction error rate with hyperparameter tuning and with all the features.
- Overall, the Gradient Boosting algorithm (R2 value: 0.6363) and XGBoost algorithm (R2 value: 0.6374) performed better. The MSE value and MAE value of both the models are approximately similar (and lowest in comparison with other models).
- The regression algorithms obtained improved results with hyperparameter tuning and Gridsearch.
- Since a generalized model has high R2 score and minimum residual error, we can further improvise these models by training the model with large amount of data, to improve the accuracy of the model.

DEPLOYMENT

Website Link: **https://loan-interest-rate-predict.herokuapp.com/**

VISUALIZATION (using MS POWER BI Tool)

LIR Dashboard final.pptx

LIR Dashboard.pbix