MACHINE LEARNING

BUSINESS REPAYMENT DATE PREDICTION

MADE WITH JUPYTER

OUR GROUP:

1	2	3	4	5
TEAM LEAD	RESEARCH	DEBUGGING	QUALITY ANALYST	TESTING
VISMAY AGARWAL	AAYUSH MARWAH	PUSHKAR PANDEY	NADEEM AHMED	SHIVANSH KUMAR SINGH
CSE/074/19	CSE/091/19	CSE/101/19	CSE/088/19	CSE/117/19

Made possible with the esteem guidance of:

Prof. Tapan Kumar Dey
&
Prof. Rahul Kumar

Objective and Use Case:

In the business industry, manufacturers provide goods in loan in exchange for the promise of repayment on its sale.

If the borrower repays the loan, then the lender would make earning. However, if the borrower fails to repay the loan, then the lender loses money or have to cease manufacturing operation.

Therefore, lenders face the problem of predicting the risk of a borrower being unable to repay a loan or delay in repayment.

In this study, the data from Lending club is used to train several Machine Learning models to determine if the borrower has the ability to repay its loan within promised time.

In addition,we would analyze the performance of the models (Random Forest, Logistic Regression, Sup-port Vector Machine, and K Nearest Neighbors).

As a result, logistic regression model is found as the optimal predictive model and it is expected that Fico Score and annual income significantly influence the forecast.

Methodology

- Null Implification
- Pre-Processing
- Feature Engineering
- EDA (Exploratory Data Analysis)
- Encoding
- Feature Selection
- Linear Regression
- Decision Tree

NULL IMPLIFICATION

- Finds empty columns in data set.
- Returns true for parameter if null column is found, else returns false.
- If a null column parameter is found, we drop that parameter i.e, remove it from data set.

```
In [4]: # now we will check empty fields
        df.isnull().sum() == df.shape[0]

Out[4]: business_code               False
        cust_number                 False
        name_customer               False
        clear_date                  False
        buisness_year               False
        doc_id                      False
        posting_date                False
        document_create_date        False
        document_create_date.1      False
        due_in_date                 False
        invoice_currency            False
        document type               False
        posting_id                  False
        area_business                True
        total_open_amount           False
        baseline_create_date        False
        cust_payment_terms          False
        invoice_id                  False
        isOpen                      False
        dtype: bool

In [5]: # As area_business is null we will remove that coll
        df.drop(['area_business'], axis=True, inplace = True)
```

PRE-PROCESSING

- Converts non readable, i.e, non int or float target and associate data frames into readable dataframes
- We do so by using regular expressions.
- Here our dataframe is date, month and year.
- Using datetime library of python we implement pre-processing.

Pre-processing

```
df["baseline_create_date"] = df["baseline_create_date"].astype(int)
df["due_in_date"] = df["due_in_date"].astype(int)
df["posting_id"] = df["posting_id"].astype(np.int64)
df["buisness_year"] = df["buisness_year"].astype(np.int64)
df['doc_id'] = df['doc_id'].fillna(0).astype(np.int64)

# converting dates in date-time format
import datetime
df['due_in_date'] = pd.to_datetime(df['due_in_date'], format='%Y%m%d')
df['clear_date'] = pd.to_datetime(df['clear_date'])
df['posting_date'] = pd.to_datetime(df['posting_date'])
df['baseline_create_date'] = pd.to_datetime(df['baseline_create_date'], format='%Y%m%d')
df['document_create_date'] = pd.to_datetime(df['document_create_date'], format='%Y%m%d')
df['document_create_date.1'] = pd.to_datetime(df['document_create_date.1'], format='%Y%m%d')


df.dtypes
```

FEATURE ENGINEERING

- After pre processing, we have date, month and year of due_in_date, posting_date and clear_date.
- Instead of storing all 3 in new columns, we will find the difference in days between all 3 results to establish a relationship b/w them.
- This saves useless storage that would have been occupied otherwise.

Feature engg

```
# We can split all dates according to there respective col days/year/month but thats nt very effective and its going to crea
# more useless col... so we will use this delay aproach instead
# we will create col in which duration from due_in_date and respected document dates are made
df['posting_date_delay'] = (df['due_in_date'] -  df['posting_date'])
df['document_create_date_delay'] = (df['due_in_date'] -  df['document_create_date'])
df['document_create_date_delay.1'] = (df['due_in_date'] -  df['document_create_date.1'])
df['baseline_create_date_delay'] = (df['due_in_date'] -  df['baseline_create_date'])

# Convert target from days format to known format(ML)
df['posting_date_delay'] = (df['posting_date_delay']).dt.days
df['document_create_date_delay'] = (df['document_create_date_delay']).dt.days
df['document_create_date_delay.1'] = (df['document_create_date_delay.1']).dt.days
df['baseline_create_date_delay'] = (df['baseline_create_date_delay']).dt.days

# we can now remove posting_date col now
df.drop(['posting_date','document_create_date','document_create_date.1','baseline_create_date'], axis=True, inplace = True)
df.head(5)
```

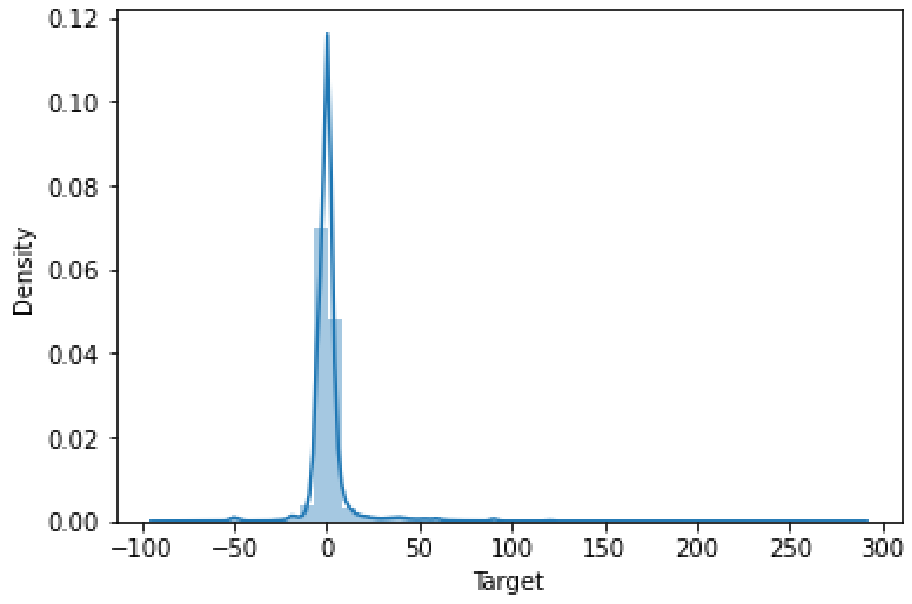
EDA

- We use displot and boxplot on training data set to find outliers by visualizing using these graphs.
- After identifying outliers, we use scatterplot to check if there are any trend in the data set.
- If we find any unique dataframes, we can drop it since it won't affect the prediction.

Applying EDA

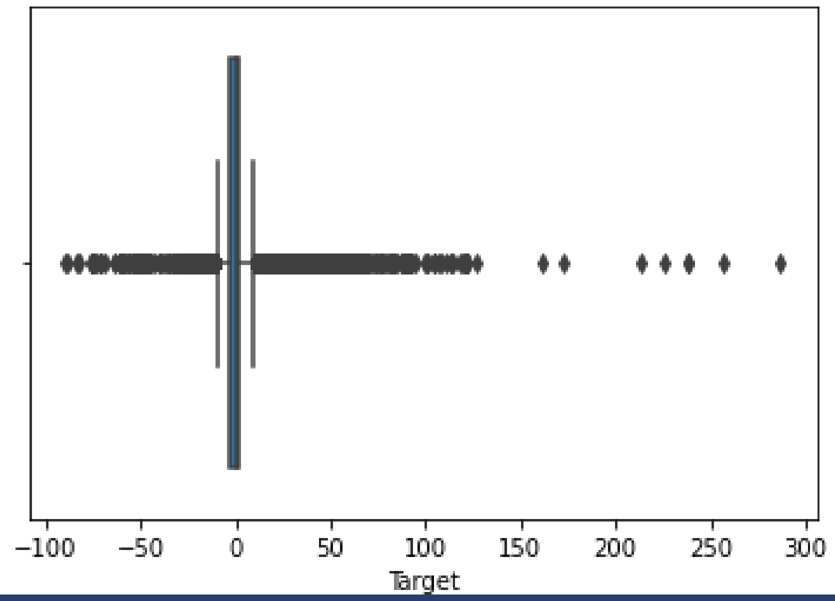
```
# plot of target values to find wether it had outliers or not
sns.distplot(y_train)
```

<AxesSubplot:xlabel='Target', ylabel='Density'>



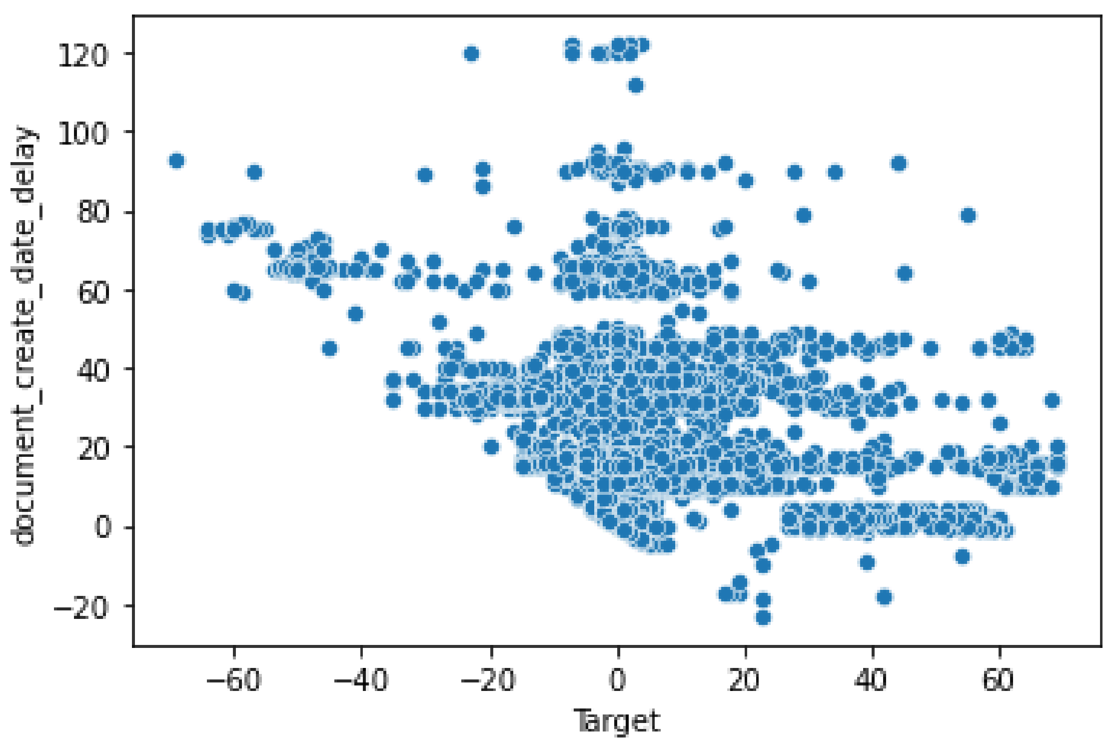

```
sns.boxplot(y_train)
```

<AxesSubplot:xlabel='Target'>



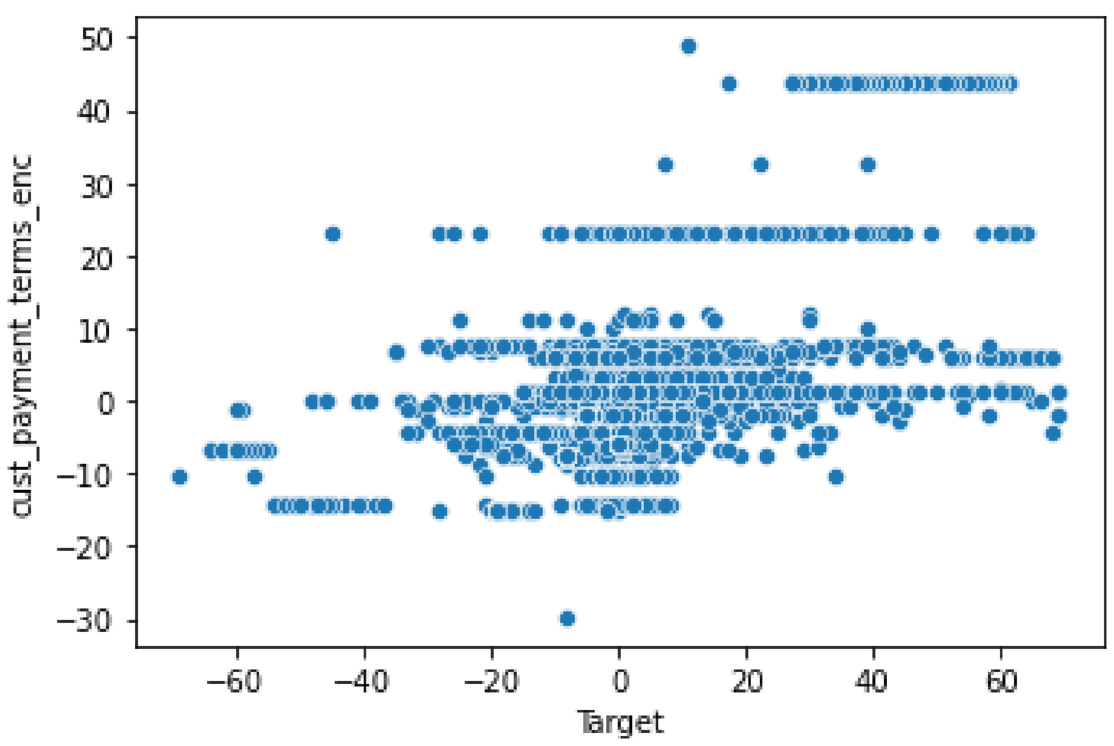
```python
sns.scatterplot(data=X_train.merge(y_train,on = X_train.index), x="Target", y="document_create_date_delay")
```

<AxesSubplot:xlabel='Target', ylabel='document_create_date_delay'>




```python
sns.scatterplot(data=X_train.merge(y_train,on = X_train.index), x="Target", y="cust_payment_terms_enc")
```

<AxesSubplot:xlabel='Target', ylabel='cust_payment_terms_enc'>



LABEL ENCODING

- Now that we have all necessary dataframes, we will try and convert these dataframes into machine readable data i.e, into integer type.
- This is done using encoding. We have implemented label encoding here.
- We are applying label encoding on data frames that are in object type.

```
# now we will apply label encoding on  document type, business_code, cust_payment_terms
from sklearn.preprocessing import LabelEncoder

# business_code
business_code_encoder = LabelEncoder()
business_code_encoder.fit(X_train['business_code'])

X_train['business_code_enc'] = business_code_encoder.transform(X_train['business_code'])
X_test['business_code_enc'] = business_code_encoder.transform(X_test['business_code'])
X_val['business_code_enc'] = business_code_encoder.transform(X_val['business_code'])

business_code_encoder.classes_
```

```
array(['CA02', 'U001', 'U002', 'U005', 'U007', 'U013'], dtype=object)
```

```
# On doc type :
document_type_encoder = LabelEncoder()
document_type_encoder.fit(X_train['document type'])

X_train['doc_type_enc'] = document_type_encoder.transform(X_train['document type'])


document_type_encoder.classes_
```

```
array(['RV', 'X2'], dtype=object)
```

Before label encoding:

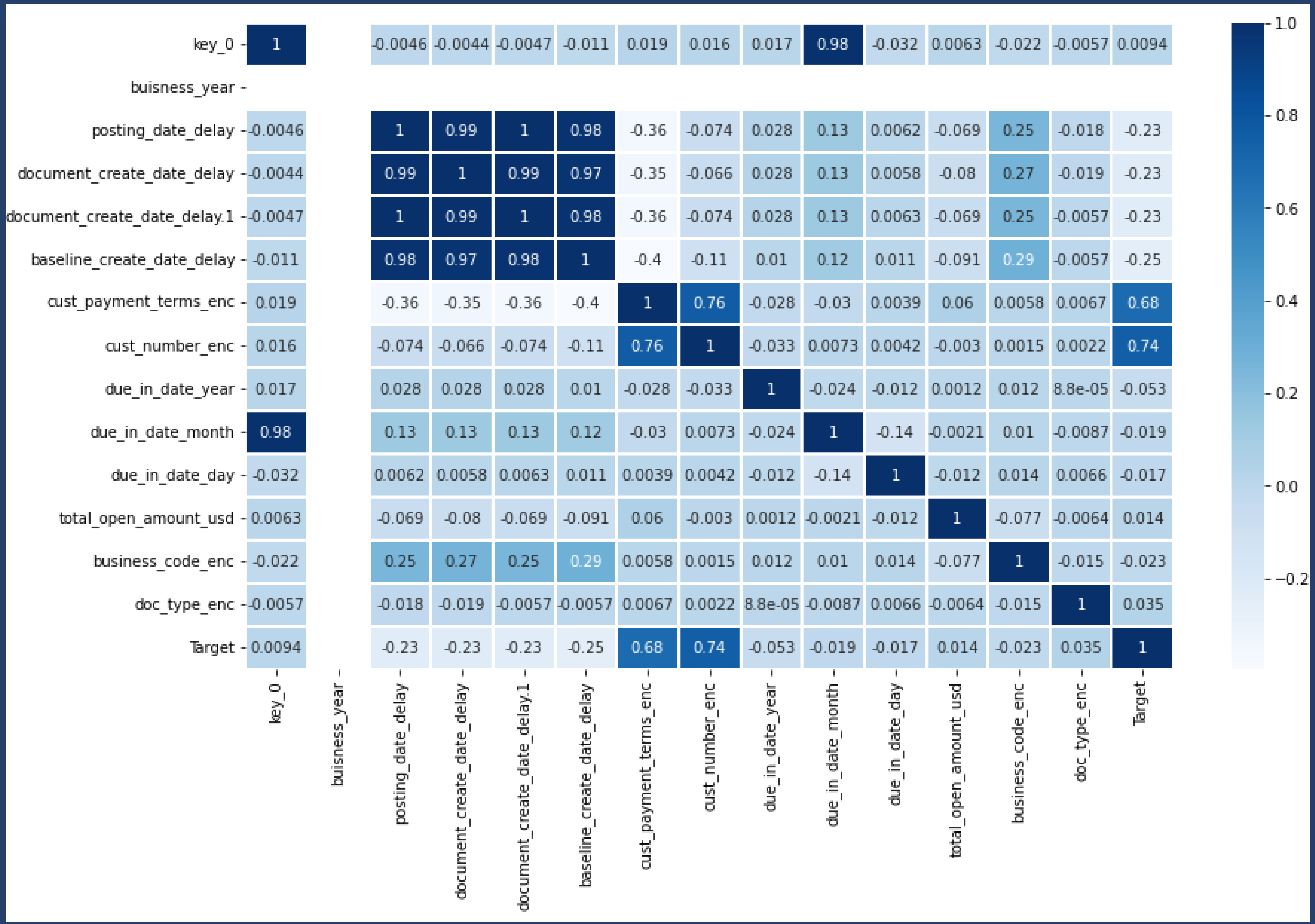
```
business_code                    object
buisness_year                     int64
document type                    object
posting_date_delay                int64
document_create_date_delay        int64
document_create_date_delay.1      int64
baseline_create_date_delay        int64
cust_payment_terms_enc          float64
cust_number_enc                 float64
due_in_date_year                  int64
due_in_date_month                 int64
due_in_date_day                   int64
total_open_amount_usd           float64
business_code_enc                 int32
doc_type_enc                      int32
dtype: object
```

After label encoding:

```
buisness_year                     int64
posting_date_delay                int64
document_create_date_delay        int64
document_create_date_delay.1      int64
baseline_create_date_delay        int64
cust_payment_terms_enc          float64
cust_number_enc                 float64
due_in_date_year                  int64
due_in_date_month                 int64
due_in_date_day                   int64
total_open_amount_usd           float64
business_code_enc                 int32
doc_type_enc                      int32
dtype: object
```

FEATURE SELECTION

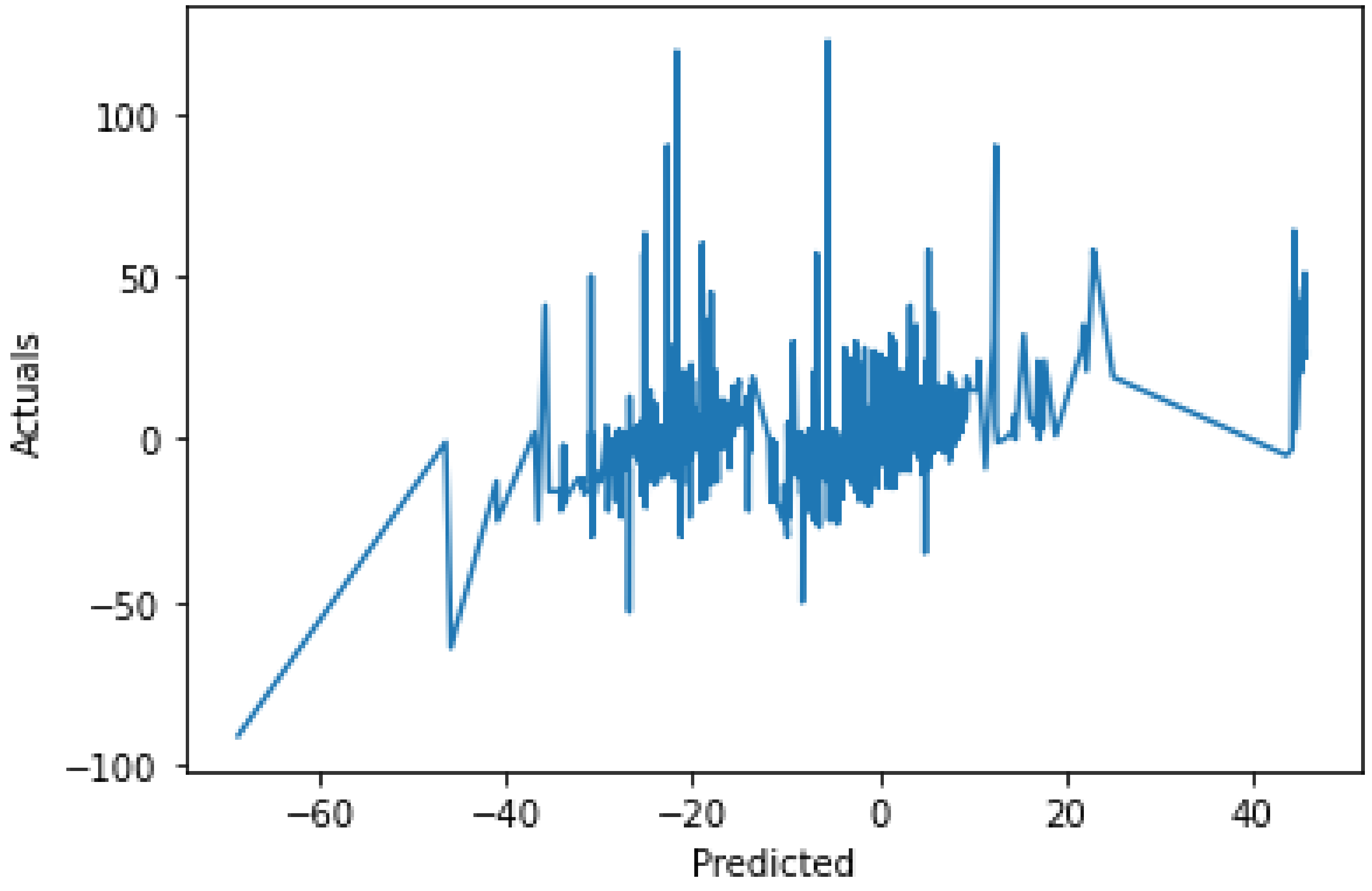
- We will now search for co-relation in train dataset using heatmap.
- We will drop highly corelated items that have a corelation of over 90%.
- Next we will find variance in these corelation and drop unique dataframes.



LINEAR REGRESSION

- Using linear regression algorithm, we will now find accuracy between train dataset and prediction data set.
- Through the lineplot graph we can see the disturbance between the two data set.
- According to the graph and result, the accuracy is only 59.99%.

```
<AxesSubplot:xlabel='Predicted', ylabel='Actuals'>
```



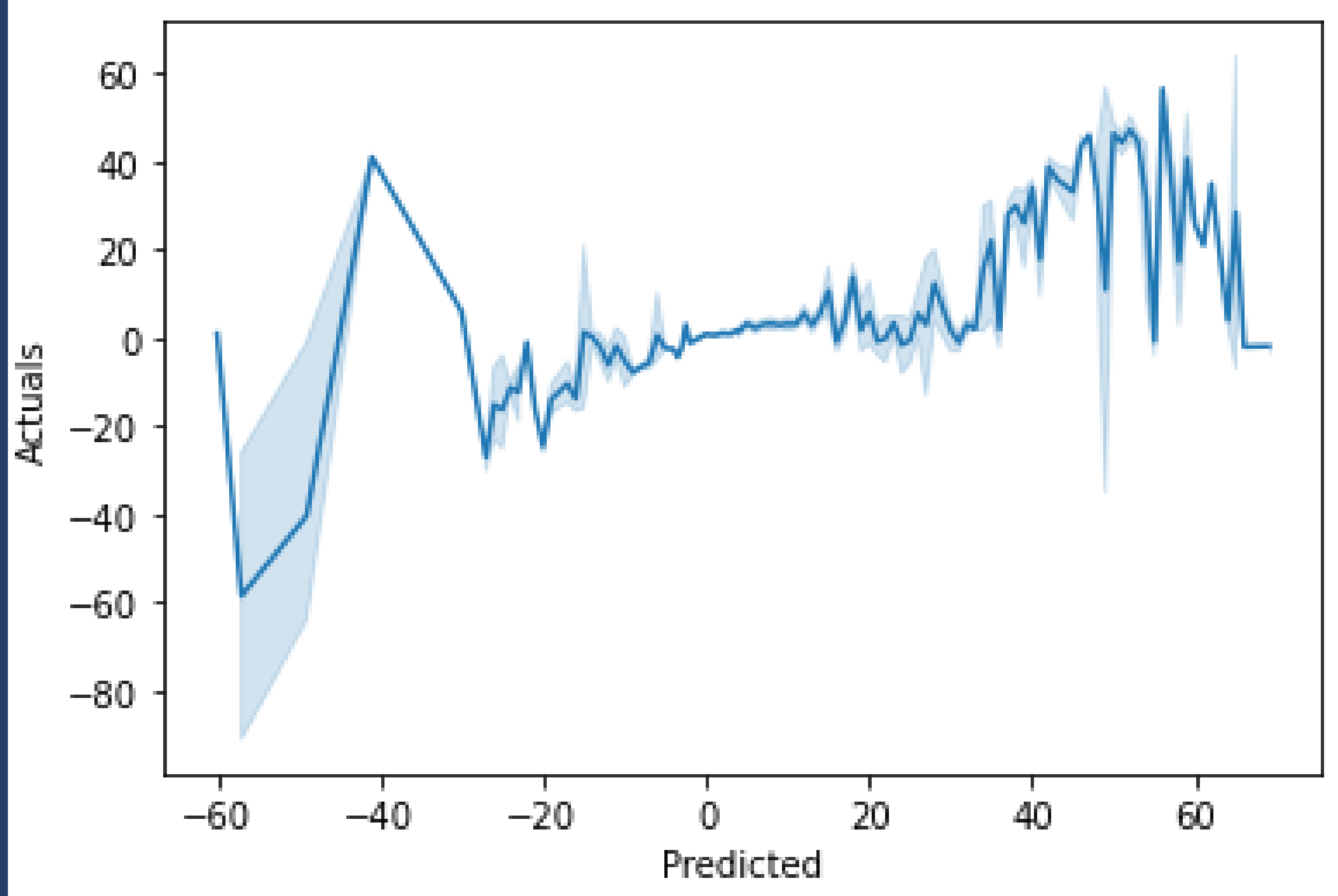

```
a = round(base_model.score(X_train,y_train)*100,2)
print(round(a,2),'%')
```

```
59.99 %
```

DECISION TREE - RANDOM FOREST ALGO

- Using decision tree regression algorithm, we will now find accuracy between train dataset and prediction data set.
- Through the lineplot graph we can see the disturbance between the two data set is better than in linear regression.
- According to the graph and result, the accuracy is over 99%.
- Hence we will use decision tree algorithm and our project is a success.

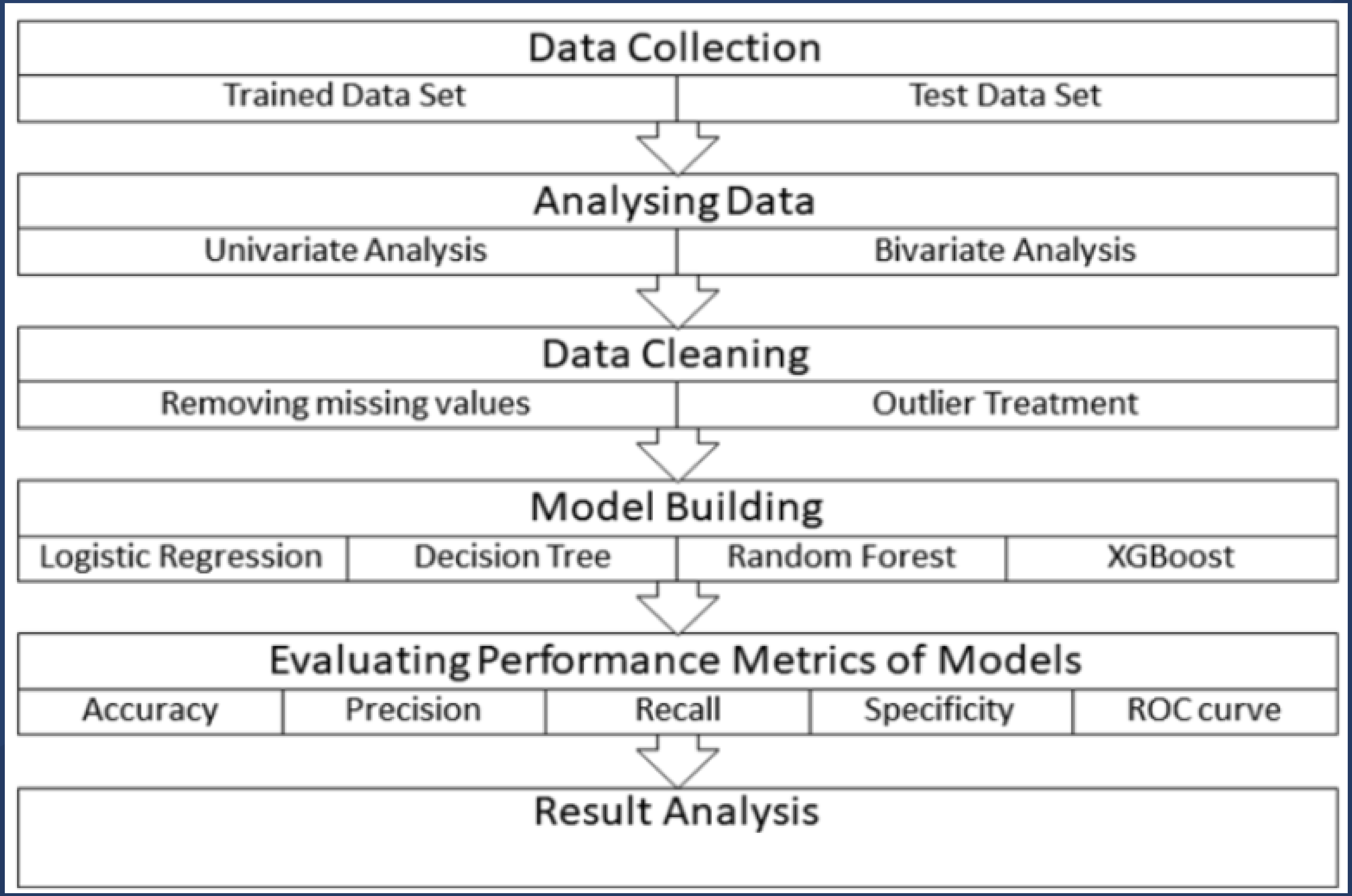
```
<AxesSubplot:xlabel='Predicted', ylabel='Actuals'>
```




```
y_predict2 = regressor.predict(X_val)
rmse = (mean_squared_error(y_val, y_predict2, squared=False))
a = round(regressor.score(X_train,y_train)*100,2)
rmse, round(a,2)
```

```
(7.715579605628708, 99.97)
```

Graphical Representation:



CONCLUSION

- We did Exploratory data Analysis on the features of this dataset and saw how each feature is distributed.
- We did bivariate and multivariate analysis to see imapct of one another on their features using charts.
- We analysed each variable to check if data is cleaned and normally distributed.
- We cleaned the data and removed NA values
- We also generated hypothesis to prove an association among the Independent variables and the Target variable. And based on the results, we assumed whether or not there is an association.
- We calculated correaltion between independent variables and found that applicant clear_date and posting_date have significant relation.
- We created dummy variables for constructing the model
- We constructed models taking different variables into account and found through odds ratio that due_in_date, posting_date & baseling_create_date are creating the most impact on target decision
- Finally, we got a model with days count as independent variable with highest accuracy in random forest algorithm.
- We tested the data and got the accuracy of 99.97 %

Thank you