



Credit Risk Management

Sathya Narayanan K

Details of the project :

Dataset Overview

Train Data Volume	80K
Test Data Volume	20K
Type of Problem	Classification



TABLE OF CONTENTS



01

About the project

To assess the credit risk for the profile and decide whether the loan would be Fully Paid or Charged-off

02

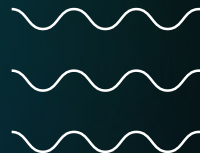
Major requirements

To understand more on the different fields in the dataset and analyze the same to apply the preprocessing and modeling

03

Project goals

To find the result of test data and compare with the Actual Data



Understanding the Data

```
data.describe()
```

✓ 0.1s

Python

	annual_inc	emp_length	fico_range_high	fico_range_low	int_rate	loan_amnt	num_actv_bc_tl	mort_acc	tot_cur_bal	open_acc	
count	8.000000e+04	80000.000000	80000.000000	80000.000000	80000.000000	80000.000000	80000.000000	80000.000000	8.000000e+04	80000.000000	80000
mean	7.604614e+04	6.196537	699.987975	695.987813	13.232898	14403.867813	3.633262	1.670701	1.414381e+05	11.605675	0
std	6.902006e+04	3.698757	31.734840	31.734075	4.771705	8703.826298	2.208597	1.971422	1.554119e+05	5.483362	0
min	0.000000e+00	0.000000	664.000000	660.000000	5.310000	750.000000	0.000000	0.000000	0.000000e+00	1.000000	0
25%	4.600000e+04	3.000000	674.000000	670.000000	9.750000	7925.000000	2.000000	0.000000	3.112450e+04	8.000000	0
50%	6.500000e+04	7.000000	694.000000	690.000000	12.740000	12000.000000	3.000000	1.000000	9.302100e+04	11.000000	0
75%	9.000000e+04	10.000000	714.000000	710.000000	15.990000	20000.000000	5.000000	3.000000	2.036260e+05	14.000000	0
max	7.141778e+06	10.000000	850.000000	845.000000	30.990000	40000.000000	32.000000	32.000000	5.172185e+06	80.000000	24



Understanding the Data

Check for Duplicates

+ Code

+ Markdown

```
data.duplicated().sum()
```

[147] ✓ 0.3s

Python

... 0

Inference : No any Duplicates Found



```
data.isna().sum()
```

[148]

✓ 0.1s

```
...   addr_state           0
      annual_inc          0
      earliest_cr_line     0
      emp_length          5846
      emp_title            6396
      fico_range_high       0
      fico_range_low        0
      grade                0
      home_ownership         0
      application_type       0
      initial_list_status    0
      int_rate              0
      loan_amnt             0
      num_actv_bc_tl        4959
      mort_acc              3475
      tot_cur_bal           4959
      open_acc              0
      pub_rec               0
      pub_rec_bankruptcies   42
      purpose               0
```

NA Check



Fixing NA

```
data['emp_length'] = data['emp_length'].fillna(data['emp_length'].mode()[0])
```

✓ 0.0s

Title of the column has large number of values and looks it is not significant with the result data hence we are dropping the column from the dataset

```
data.drop(columns="emp_title",inplace=True)
```

✓ 0.0s

Python

To fill na for column No. of Active Banks we can go with group by emp_length and then take the mean to arrive at the No. of Active Cards

```
data['num_actv_bc_tl'] = data.groupby(['emp_length'])['num_actv_bc_tl'].transform(lambda x : x.fillna(x.mean()))
```

162] ✓ 0.0s

Python

```
data['mort_acc'] = data.groupby(['emp_length'])['mort_acc'].transform(lambda x : x.fillna(x.mean()))
```

✓ 0.0s

```
data['tot_cur_bal'] = data.groupby(['emp_length'])['tot_cur_bal'].transform(lambda x : x.fillna(x.mean()))
```

✓ 0.0s

```
data['pub_rec_bankruptcies'] = data['pub_rec_bankruptcies'].fillna(data['pub_rec_bankruptcies'].mode()[0])
```

✓ 0.0s

```
data['revol_util'] = data.groupby(['emp_length'])['revol_util'].transform(lambda x : x.fillna(x.mean()))
```

✓ 0.0s

[+ Code](#) [+ Markdown](#)

```
data['title'] = data['title'].fillna(data['title'].mode()[0])
```

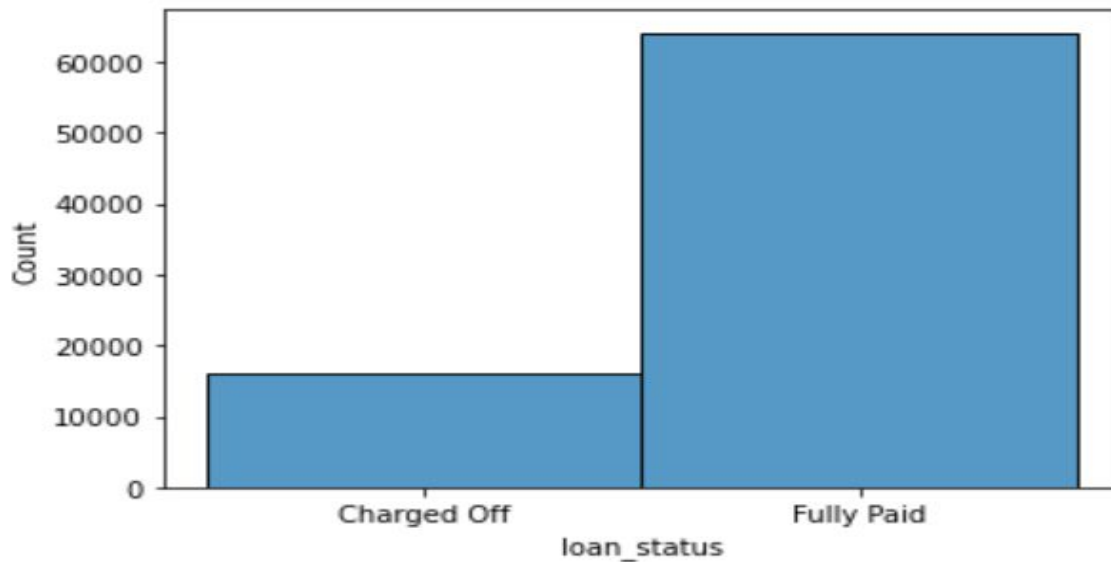
✓ 0.0s

Analysis of the Input Data

```
sns.histplot(data['loan_status'])
```

✓ 0.3s

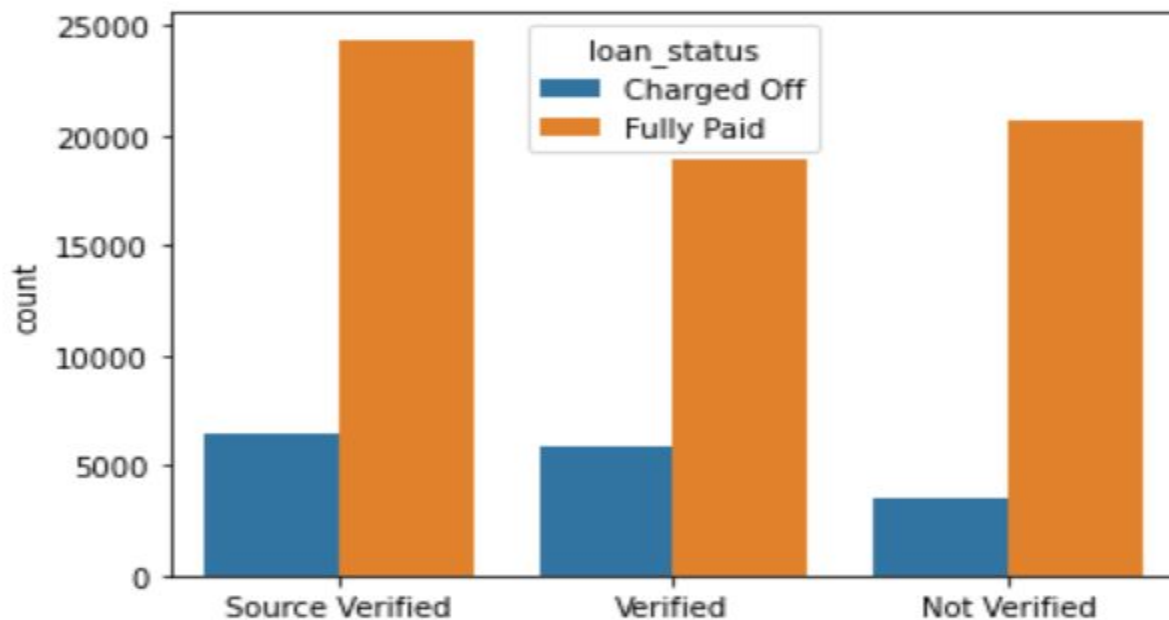
<AxesSubplot:xlabel='loan_status', ylabel='Count'>



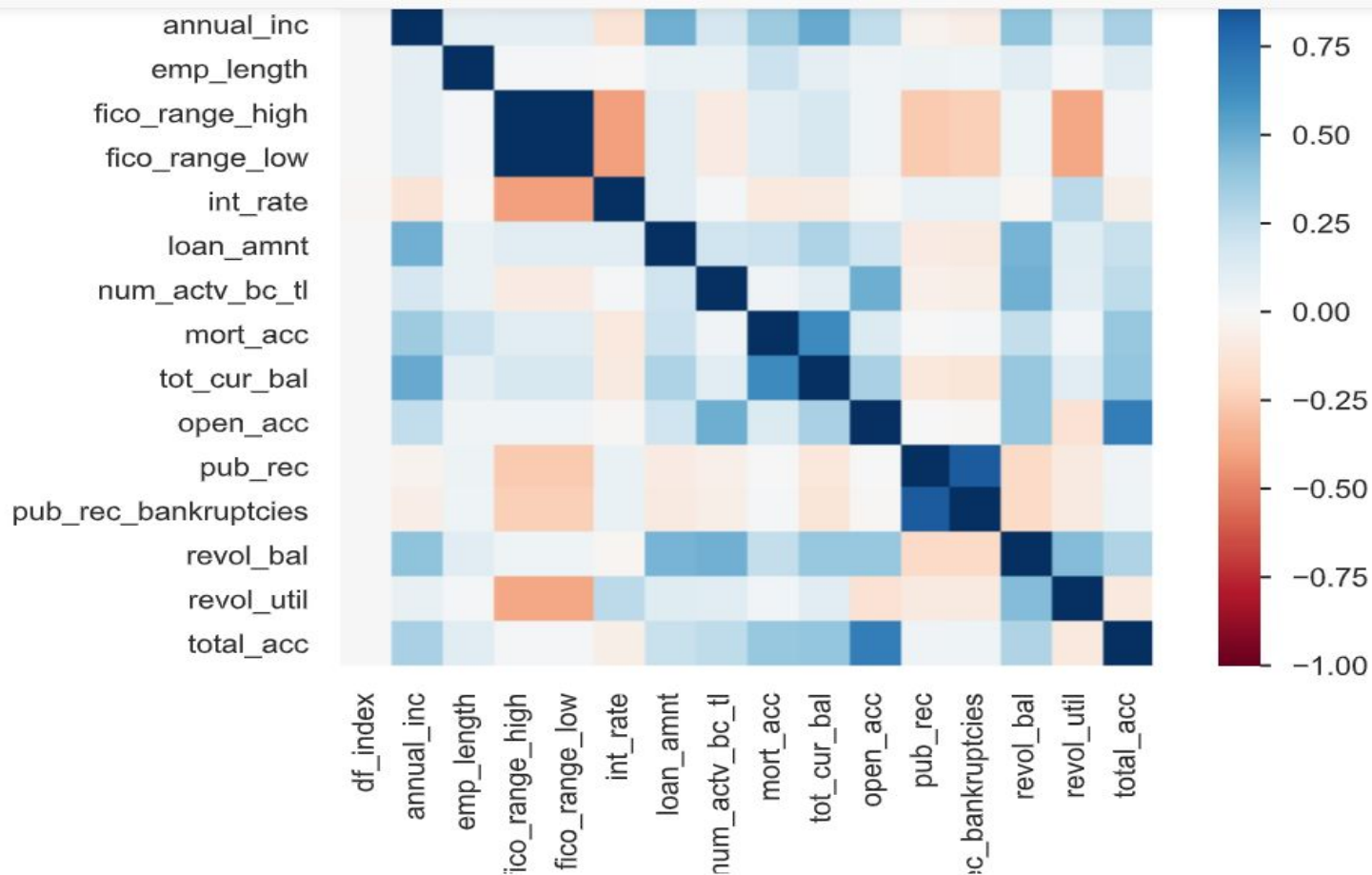
```
sns.countplot(data=data,x='verification_status',hue='loan_status')
```

✓ 0.2s

```
<AxesSubplot:xlabel='verification_status', ylabel='count'>
```



Analysis of the Input Data



```
data.drop(columns="title",inplace=True)
```

2] ✓ 0.1s

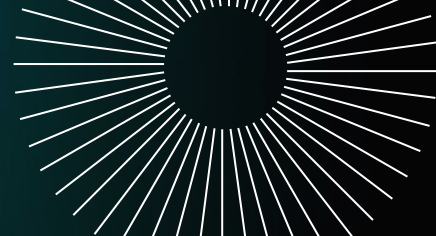
```
data.drop(columns="addr_state",inplace=True)
```

3] ✓ 0.0s

```
data['earliest_cr_line'].value_counts()
```

4] ✓ 0.0s

```
Oct-2001    701
Aug-2001    686
Sep-2003    683
Sep-2004    680
Sep-2002    668
...
Jul-2015     1
May-1959     1
Nov-1961     1
Jun-2015     1
Dec-1959     1
Name: earliest_cr_line, Length: 644, dtype: int64
```



Modification of `earliest_cr_line` Column

```
data['new_earliest_cr_line'] = pd.to_datetime(data["earliest_cr_line"])
```

✓ 0.0s

```
data['new_earliest_cr_line'].dtypes
```

✓ 0.0s

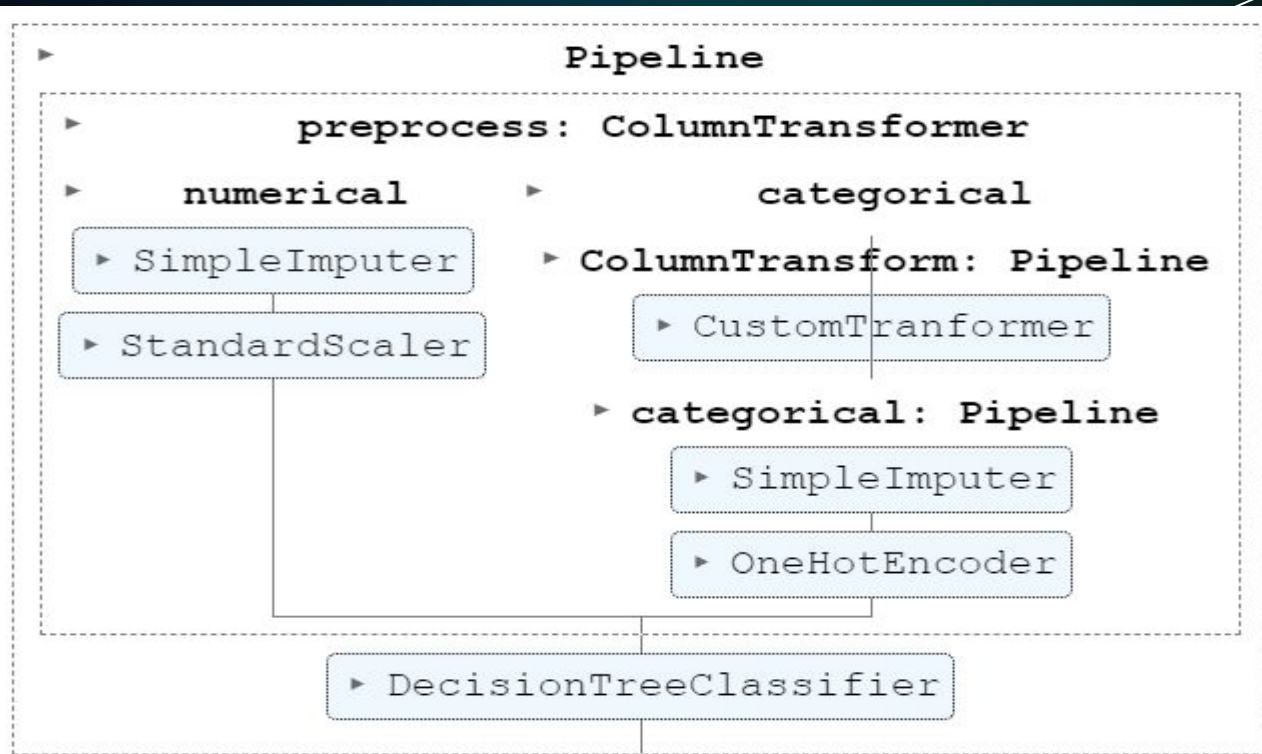
```
dtype('<M8[ns]')
```

```
data['months'] = (pd.to_datetime('today') - data['new_earliest_cr_line'])/np.timedelta64(1, 'M')
```

✓ 0.0s

```
data['months']
```

✓ 0.0s



Anal Result by comparing the Results with the Actuals

Note : Algorithm Used : Decision Tree

For Categorical Columns used Simple Imputer and One Hot Encoding

For Numerical Columns used Simple Imputer and Standard Scaler

Total matching Records ==13974

Total matching Records with Fully Paid==12781

Total matching Records with Charged Off==1193



Analysis of data Multi Model Run Comparison

```
pipe_lr=Pipeline(steps=[
    ('prepro',preprocess_full),
    ('lr',LogisticRegression())
])

pipe_dt=Pipeline(steps=[
    ('prepro',preprocess_full),
    ('dt',DecisionTreeClassifier())
])

pipe_knn=Pipeline(steps=[
    ('prepro',preprocess_full),
    ('knn',KNeighborsClassifier())
])

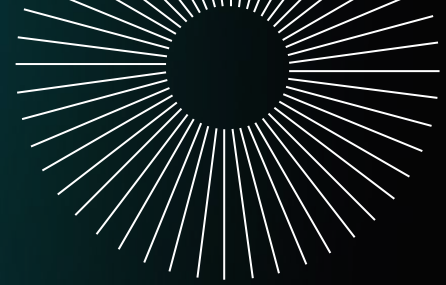
pipe_rf=Pipeline(steps=[
    ('prepro',preprocess_full),
    ('rf',RandomForestClassifier())
])
```

✓ 0.0s

Analysis of data

Multi Model Run Comparison

	Model	Accuracy
0	Log_reg	-0.622968
1	knn	-0.629090
2	decisiontree	-0.897077
3	RandomForest	-0.599040

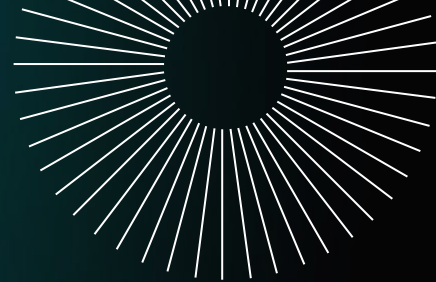


Results post sampling with Decision Tree

```
... Total matching Records ==14106  
Total matching Records with Fully Paid==12989  
Total matching Records with Charged Off==1117
```



Analysis of data



Thank you !

