DATA SCIENCE PROJECT : 1

TOPIC : LendingClub Data Analysis

Group 1:

Jinesh Bhatewara: 321006

Yash Chilwar : 321010

Anom Devgun : 321013

Atharva Gogawale: 321016

Simoni Patani(e2): 321037

Table of Contents

Aim :

Lending Club connects people who need money (borrowers) with people who have money (investors). Hopefully, as an investor you would want to invest in people who showed a profile of having a high probability of paying you back. This project aims to create a model that will help predict this.

Objective :

We are trying to classify and predict whether or not the borrower paid back their loan in full. The csv file used has been cleaned of NA values.

Algorithm and Model used :

We have used the **Decision Tree Algorithm** and the **Random Forest Model**.

Theory :

Decision Tree :

Decision tree as the name suggests it is a flow like a tree structure that works on the principle of conditions. It is efficient and has strong algorithms used for predictive analysis. It has mainly attributed that include internal nodes, branches and a terminal node.

Every internal node holds a "test" on an attribute, branches hold the conclusion of the test and every leaf node means the class label. This is the most used algorithm when it comes to supervised models. It is used for both classifications as well as regression. It is often termed as ***"CART"*** that means classification and regression tree

.

In data science, one cannot always rely on linear models because there is non-linearity at maximum places. It is noted that tree models like **Random forest,** Decision trees deal in a good way with non-linearity.

Random Forest :

Random forest on randomly selected data creates different decision trees and then makes the collection of votes from trees to compute the class of the test object.

Random forest is based on the divide-and-conquer perspective of decision trees that are created by randomly splitting the data. Generating decision trees is also known as a forest. Each decision tree is formed using feature selection indicators like information gain**,** gain ratio, and Gini index of each feature. Each tree is dependent on an independent sample. Considering it to be a classification problem, then each tree computes votes and the highest votes class is chosen. If its regression, the average of all the tree's outputs is declared as the result. It is the most powerful algorithm compared to all others.

The only difference that makes random forest algorithms different from decision trees is the computation that is made to find the root node and splitting the attributes nodes will run in a random way.

DATA SET DESCRIPTION :

We have used the ending data from 2007-2010 and tried to classify and predict whether or not the borrower paid back their loan in full. The csv file used has been cleaned of NA values.

The dataset has 9579 rows and 14 columns. Here are what the columns represent:

- credit.policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.
- purpose: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").
- int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates.
- installment: The monthly installments owed by the borrower if the loan is funded.
- log.annual.inc: The natural log of the self-reported annual income of the borrower.
- dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income).
- fico: The FICO credit score of the borrower.
- days.with.cr.line: The number of days the borrower has had a credit line.
- revol.bal: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).
- revol.util: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).
- inq.last.6mths: The borrower's number of inquiries by creditors in the last 6 months.
- delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.
- pub.rec: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).

CODE :

import pandas as pd

import numpy as np

import seaborn as sns

sns.set_style("darkgrid")

import matplotlib.pyplot as plt

%matplotlib inline

loans = pd.read_csv("loan_data.csv")

loans.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
credit.policy        9578 non-null int64
purpose              9578 non-null object
int.rate             9578 non-null float64
installment          9578 non-null float64
log.annual.inc       9578 non-null float64
dti                  9578 non-null float64
fico                 9578 non-null int64
days.with.cr.line    9578 non-null float64
revol.bal            9578 non-null int64
revol.util           9578 non-null float64
inq.last.6mths       9578 non-null int64
delinq.2yrs          9578 non-null int64
pub.rec              9578 non-null int64
not.fully.paid       9578 non-null int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

loans.describe()

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs
count	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9.578000e+03	9578.000000	9578.000000	9578.00000
mean	0.804970	0.122640	319.089413	10.932117	12.606679	710.846314	4560.767197	1.691396e+04	46.799236	1.577469	0.16370
std	0.396245	0.026847	207.071301	0.614813	6.883970	37.970537	2496.930377	3.375619e+04	29.014417	2.200245	0.54621
min	0.000000	0.060000	15.670000	7.547502	0.000000	612.000000	178.958333	0.000000e+00	0.000000	0.000000	0.00000
25%	1.000000	0.103900	163.770000	10.558414	7.212500	682.000000	2820.000000	3.187000e+03	22.600000	0.000000	0.00000
50%	1.000000	0.122100	268.950000	10.928884	12.665000	707.000000	4139.958333	8.596000e+03	46.300000	1.000000	0.00000
75%	1.000000	0.140700	432.762500	11.291293	17.950000	737.000000	5730.000000	1.824950e+04	70.900000	2.000000	0.00000
max	1.000000	0.216400	940.140000	14.528354	29.960000	827.000000	17639.958330	1.207359e+06	119.000000	33.000000	13.00000

loans.head()

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	

Exploratory Data Analysis

Data visualization

We have used seaborn and pandas built-in plotting capabilities.

Created a histogram of two FICO distributions on top of each other, one for each credit.policy outcome.

```
# loans.groupby('credit.policy')['fico'].hist(bins=30, alpha=0.5, figsize=(10,8))
```

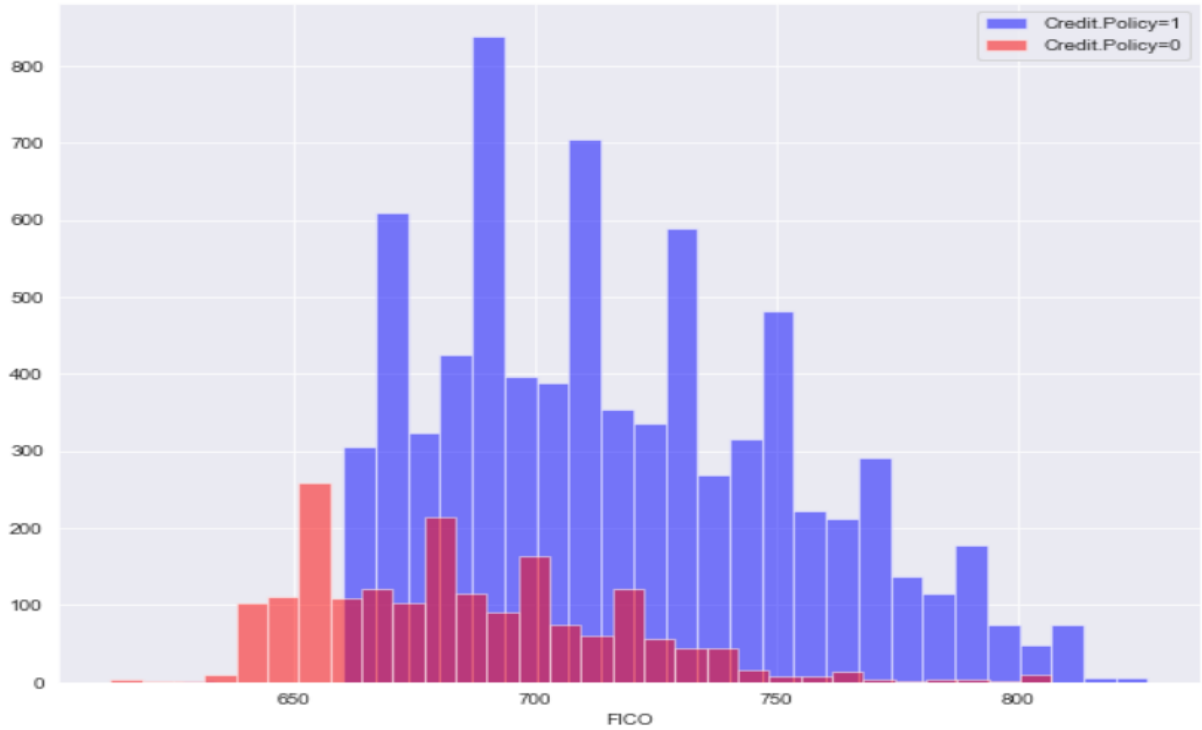
```
plt.figure(figsize=(10,8))

loans[loans['credit.policy']==1]['fico'].hist(alpha=0.5,color='blue',
                                              bins=30,label='Credit.Policy=1')

loans[loans['credit.policy']==0]['fico'].hist(alpha=0.5,color='red',
```

bins=30,label='Credit.Policy=0')

plt.legend()

plt.xlabel('FICO')

```
Text(0.5, 0, 'FICO')
```



Created a similar figure, except this time select by the not.fully.paid column.

plt.figure(figsize=(10,8))

loans[loans['not.fully.paid']==1]['fico'].hist(alpha=0.5,color='blue',

bins=30,label='Credit.Policy=1')

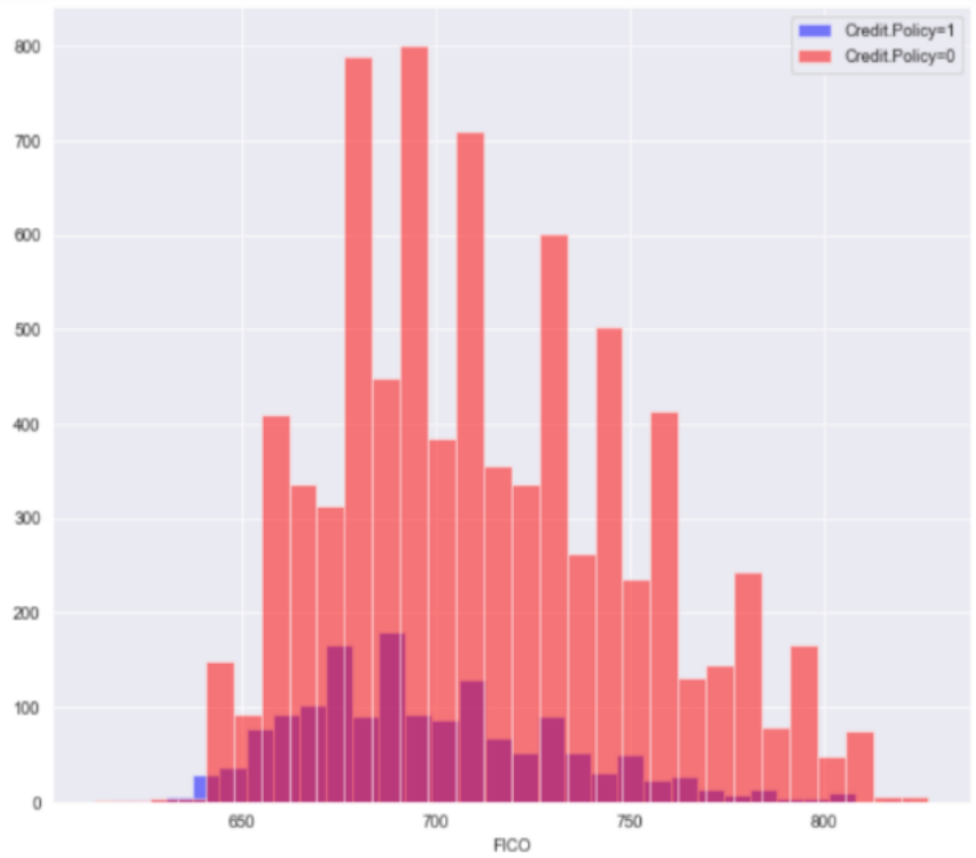
loans[loans['not.fully.paid']==0]['fico'].hist(alpha=0.5,color='red',

bins=30,label='Credit.Policy=0')

plt.legend()

plt.xlabel('FICO')

```
Text(0.5, 0, 'FICO')
```

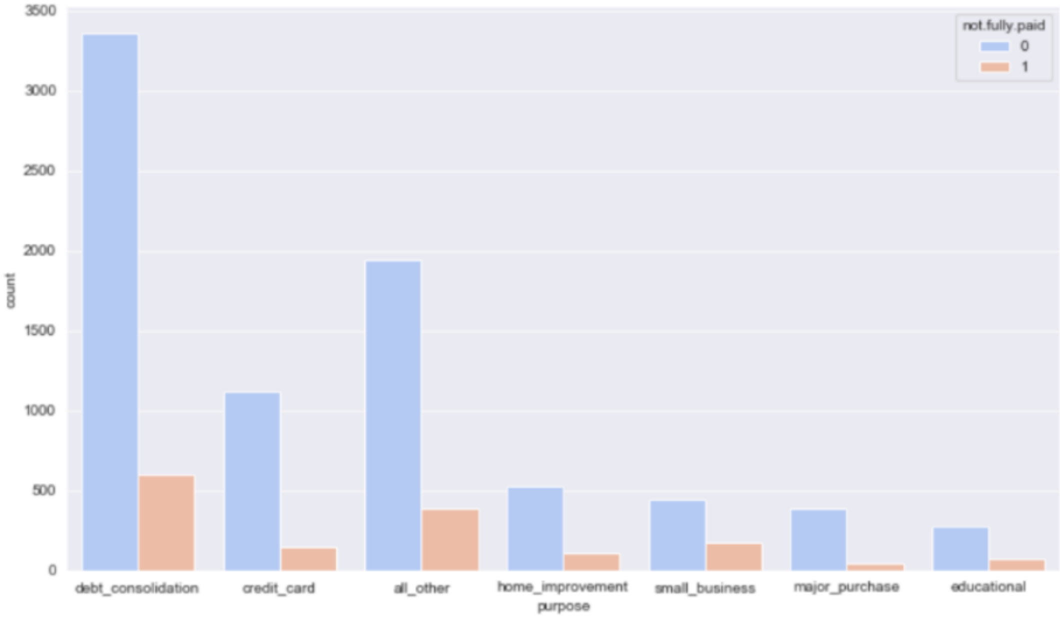


*Created a countplot using seaborn showing the counts of loans by purpose, with the color hue defined by not.fully.paid. *

plt.figure(figsize=(12,7))

sns.countplot(x= "purpose",hue= "not.fully.paid", data=loans, palette= "coolwarm")

```
<matplotlib.axes._subplots.AxesSubplot at 0x2779607fe88>
```



The trend between FICO score and interest rate.

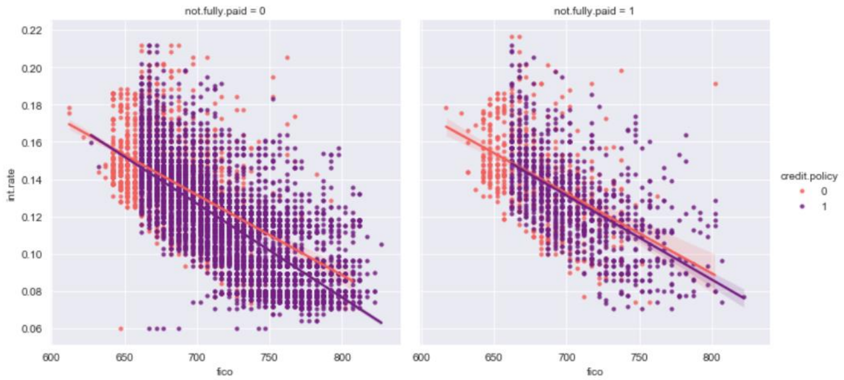
In [167]:
```
sns.jointplot(x="fico", y="int.rate", data= loans, color= "purple", s=10)
```

Out[167]:
```
<seaborn.axisgrid.JointGrid at 0x277972da448>
```

Created the following lmplots to see if the trend differed between not.fully.paid and credit.policy.

In [168]:
```
sns.lmplot(x= "fico", y="int.rate", data=loans, col="not.fully.paid", hue="credit.policy",palette= "magma_r", scatter_kws={"
```

Out[168]: <seaborn.axisgrid.FacetGrid at 0x277978ad348>



Setting up the Data

set up the data for Random Forest Classification Model!

Check loans.info()

In [169]:
```
loans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   credit.policy      9578 non-null   int64
 1   purpose            9578 non-null   object
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64
 11  delinq.2yrs        9578 non-null   int64
 12  pub.rec            9578 non-null   int64
 13  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

Categorical Features

Notice that the **purpose** column is categorical

That means we need to transform them using dummy variables so sklearn will be able to understand them.

Created a list of 1 element containing the string 'purpose'. Call this list cat_feats.

cat_feats=['purpose']

Used pd.get_dummies(loans,columns=cat_feats,drop_first=True) to create a fixed larger dataframe that has new feature columns with dummy variables. Have Set this dataframe as final_data.

final_data= pd.get_dummies(loans, columns= cat_feats, drop_first=True)

final_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 19 columns):
credit.policy                 9578 non-null int64
int.rate                      9578 non-null float64
installment                   9578 non-null float64
log.annual.inc                9578 non-null float64
dti                           9578 non-null float64
fico                          9578 non-null int64
days.with.cr.line             9578 non-null float64
revol.bal                     9578 non-null int64
revol.util                    9578 non-null float64
inq.last.6mths                9578 non-null int64
delinq.2yrs                   9578 non-null int64
pub.rec                       9578 non-null int64
not.fully.paid                9578 non-null int64
purpose_credit_card           9578 non-null uint8
purpose_debt_consolidation    9578 non-null uint8
purpose_educational           9578 non-null uint8
purpose_home_improvement      9578 non-null uint8
purpose_major_purchase        9578 non-null uint8
purpose_small_business        9578 non-null uint8
dtypes: float64(6), int64(7), uint8(6)
memory usage: 1.0 MB
```

Train Test Split

splitting the data into a training set and a testing set!

Used sklearn to split data into a training set and a testing set.

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
X = final_data.drop('not.fully.paid', axis=1)
y = final_data['not.fully.paid']
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size= 0.3, random_state= 101)
```

Training a Decision Tree Model

Import DecisionTreeClassifier

In [13]:
```
from sklearn.tree import DecisionTreeClassifier
```

Created an instance of DecisionTreeClassifier() called dtree and fit it to the training data.

In [14]:
```
dtree= DecisionTreeClassifier()
```

In [15]:
```
dtree.fit(X_train, y_train)
```

Out[15]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

Predictions and Evaluation of Decision Tree

Created predictions from the test set and create a classification report and a confusion matrix.

In [16]:
```
pred = dtree.predict(X_test)
```

In [17]:
```
from sklearn.metrics import classification_report as cr, confusion_matrix as cm
```

In [18]:
```
print(cr(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.82      0.84      2431
           1       0.20      0.24      0.21       443

    accuracy                           0.73      2874
   macro avg       0.53      0.53      0.53      2874
weighted avg       0.75      0.73      0.74      2874
```

In [19]:
```
print(cm(y_test, pred))
```

```
[[2000  431]
 [ 338  105]]
```

Checking the accuracy of Decision Tree model

[22]
```
score = accuracy_score(y_test, pred)
print("The accuracy of Decision tree is ",score*100,"%")
```

```
The accuracy of Decision tree is  72.86012526096033 %
```

The accuracy of decision tree was 72.8%

Training the Random Forest model

Created an instance of the RandomForestClassifier class and fit it to our training data from the previous step.

In [20]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [21]:
```
rfc= RandomForestClassifier()
```

In [22]:
```
rfc.fit(X_train, y_train)
```

```
D:\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators will change fro
m 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[22]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Checking the accuracy of Random Forest model:

[35]
```
score = accuracy_score(y_test,rfc_pred)
print("The accuracy of Random Forest is ",score*100,"%")
```

```
The accuracy of Random Forest is  84.41196938065414 %
```

The accuracy of Random forest was 84.4%

Predictions and Evaluation

Predicted off the y_test values and evaluate our model.

Predicted the class of not.fully.paid for the X_test data.

In [23]:
```python
rfc_pred= rfc.predict(X_test)
```

Created a classification report from the results

In [24]:
```python
from sklearn.metrics import classification_report as cr, confusion_matrix as cm
```

In [25]:
```python
print(cr(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.98      0.91      2431
           1       0.34      0.06      0.11       443

    accuracy                           0.84      2874
   macro avg       0.59      0.52      0.51      2874
weighted avg       0.77      0.84      0.79      2874
```

Confusion Matrix for the predictions.

In [26]:
```python
print(cm(y_test, rfc_pred))
```

```
[[2376   55]
 [ 415   28]]
```

As the accuracy of Random Forest was more than the decision tree we took user's input and predicted that he/she can fully pay the Loan or not i.e 1/0

```python
X_test_temp = X_test
X_test_temp
```

[38]
```python
temp = final_data.drop('not.fully.paid', axis=1).mean()
for i in range(len(temp)):
    X_test_temp[final_data.drop('not.fully.paid', axis=1).columns[i]][5244]=float(input(final_data.drop('not.fully.paid', axis=1).columns[i]))
```

[39]
```python
rfc_pred_temp = rfc.predict(X_test_temp)
rfc_pred_temp[0]
```

```
0
```

Our program takes the input from the user in a pop-up bar :

```
|
credit.policy (Press 'Enter' to confirm or 'Escape' to cancel)
```

```
0.12
int.rate (Press 'Enter' to confirm or 'Escape' to cancel)
```

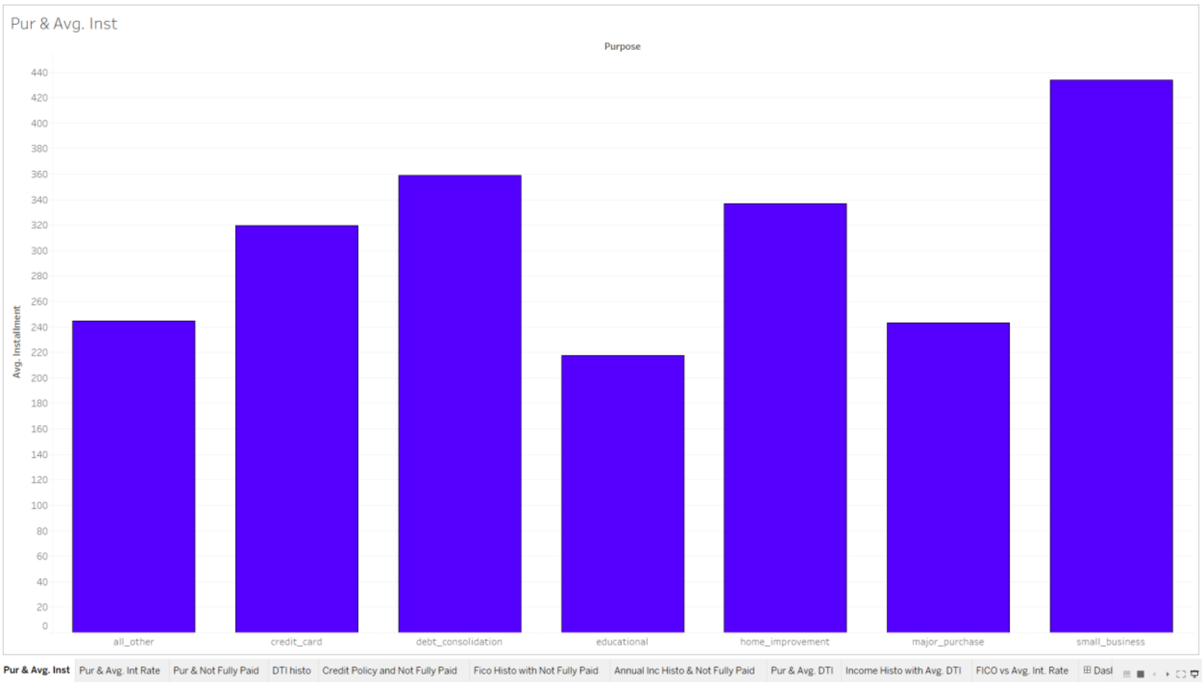
INFERENCES (based on the code) :

1. Random forest is more accurate model than decision tree for our loan data
2. The average interest rate was 12.26%
3. The average installment was 321$ pm
4. The highest monthly installment was 940$ and the lowest was only 15$
5. Maximum of 13 times a borrower has missed his deadline for payment by a month
6. The number of loans taken was highest for the purpose “debt_consolidation” which means people are taking loans to pay their other debts
7. While checking the public record most of the people i.e. 9019 had no bad records while only one person had 4,5 records but 533 had only one black spot on their records
8. People with higher FICO scores had more credit approvals
9. The people with less FICO score had less chances of clearing their defaults

DATA VISUALIZATION :

For the purpose of data visualization , the dataset was imported to the **Tableau** software , graphs were made and inferences were taken.

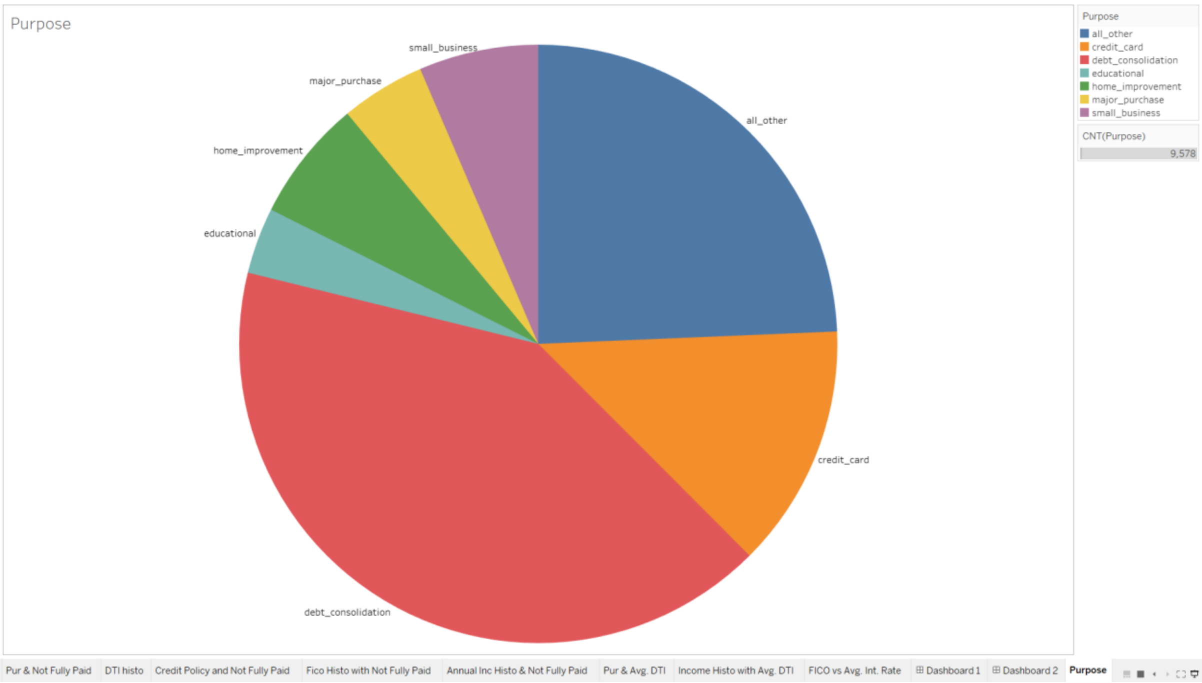
Graph 1:



Sheet 1: Bar graph of Purpose of loan and Average Installment

Highest avg. installment is done for small businesses and lowest is done for educational purpose. Installments are high for high amount of loans and low for low amount of loans. Small businesses, debt consolidation, home improvements, and credit card all required high loans so high installment.

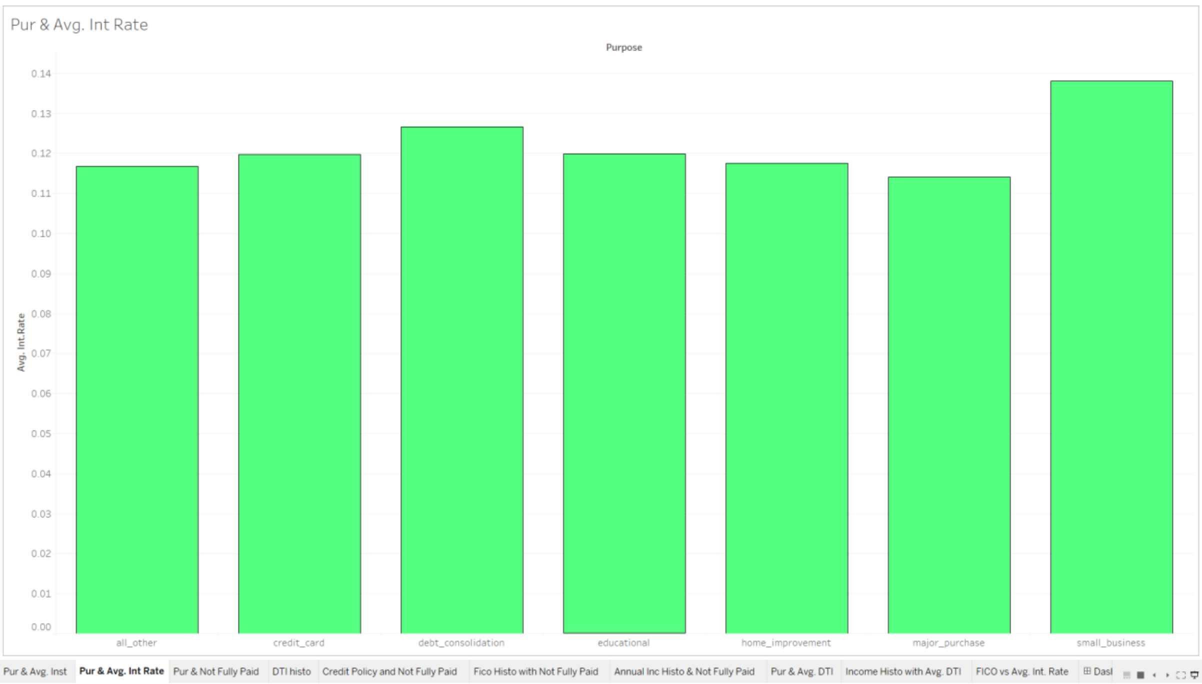
Graph 2:



Sheet 2: Pie Chart of Purpose

Most of the loans are taken for debt consolidation, about 3957 (41%). Least amount of loans are taken for Educational purpose, about 343 (3.5%). And for credit card 13%, all other purpose 24.33%, small businesses 6.46%, major purchase 4.56%, home improvement 6.56%.

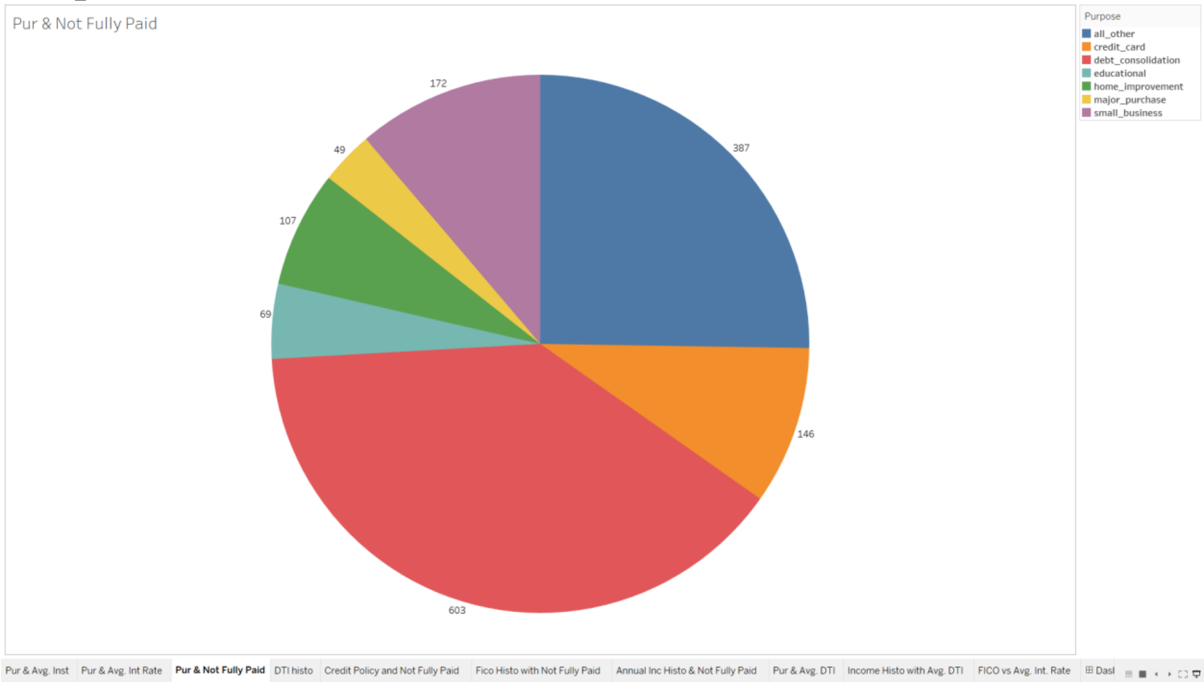
Graph 3:



Sheet 3: Bar Graph of Purpose and Average Interest Rate

Interest Rate is highest for small business purpose (0.138) and lowest for major purchase (0.114). Almost for all purposes interest rate is similar except for small business purpose.

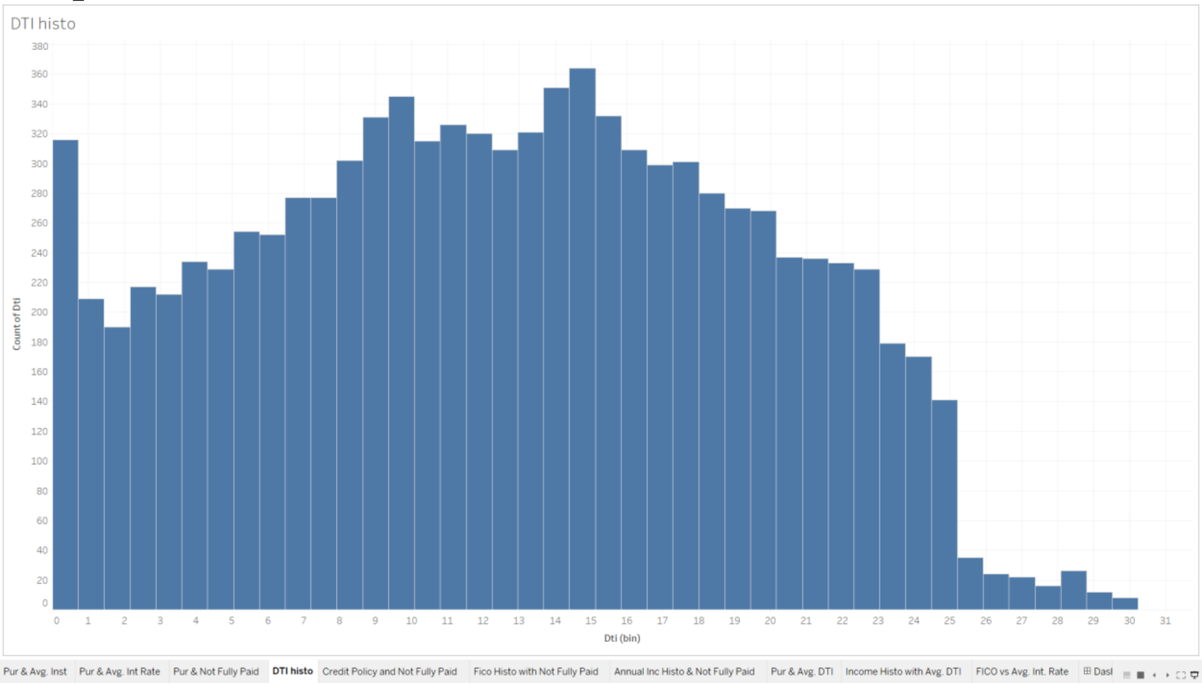
Graph 4:



Sheet 4: Pie Chart of Purpose and Not Fully Paid

Most of the people (39.33%) who took loan for debt consolidation are not able to pay the full loan. Most of the people who took for major purchase are able to pay the full loan with only few exceptions (3.19%). Loans should be given more to major purchase, home improvement, educational and small business purposes as they return full amount and should be given less and carefully for debt consolidation, credit card, all other purposes.

Graph 5:



Sheet 5: Histogram of DTI (Debt To Income ratio)

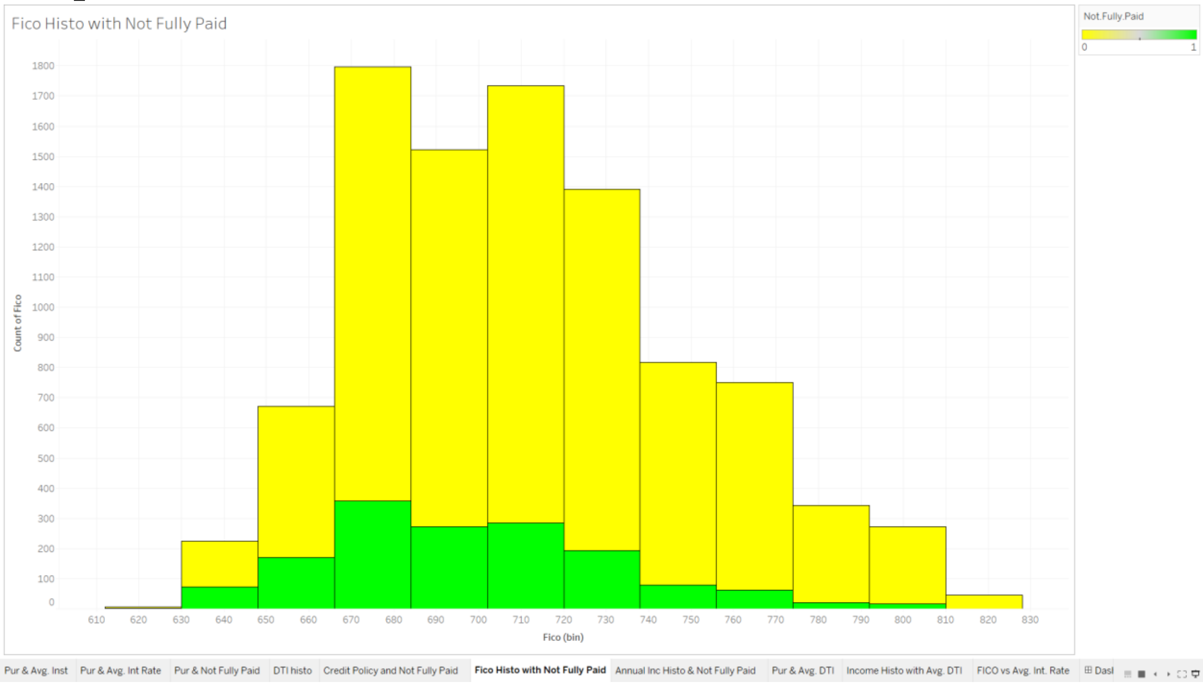
Most of the people spend 6% - 23% of their income to clear their debt. Also there are a significant amount of people who spend nothing of their income to clear their debt.

Graph 6:

Sheet 6: Packet Bubble Diagram of Credit Policy and Not Fully Paid.

Most of the people who have a credit policy are not able to pay the full loan (66.14%). The people who don't have a credit policy and are not able to pay the full loan are 33.85%.

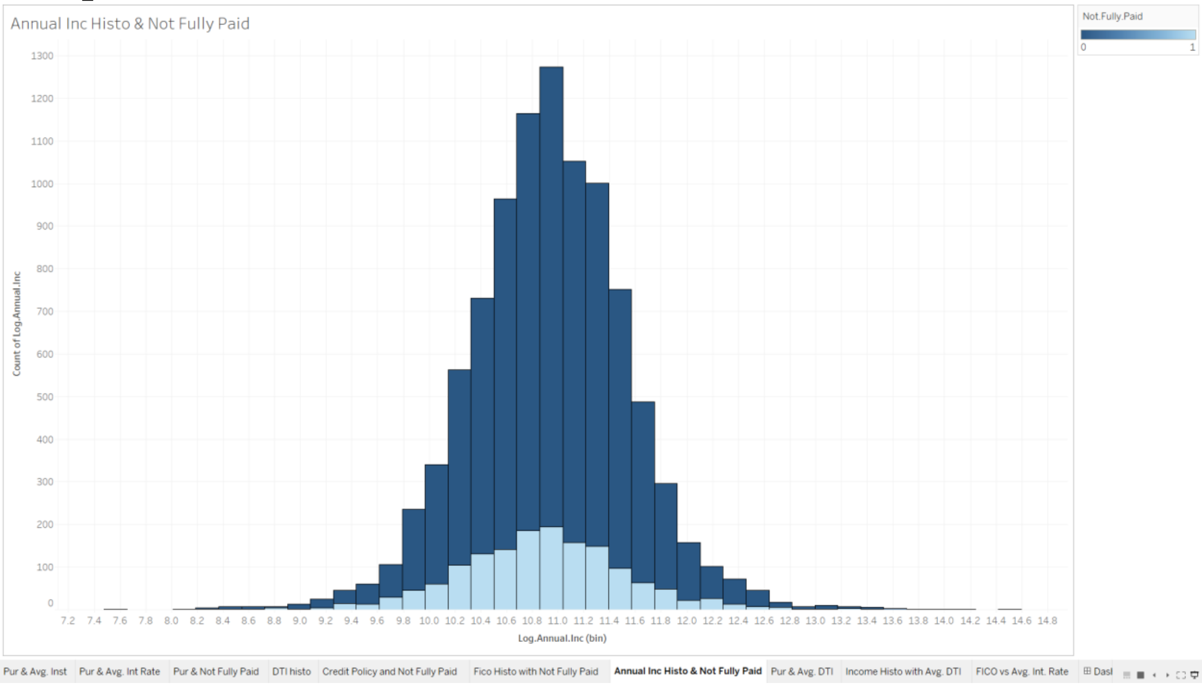
Graph 7:



Sheet 7: Divided Histogram of FICO

Most of the people have a FICO score between 648 – 720 and on an average 16.5% don't pay the full loan. Higher the FICO score higher is the probability of the person paying the full loan and vise-versa.

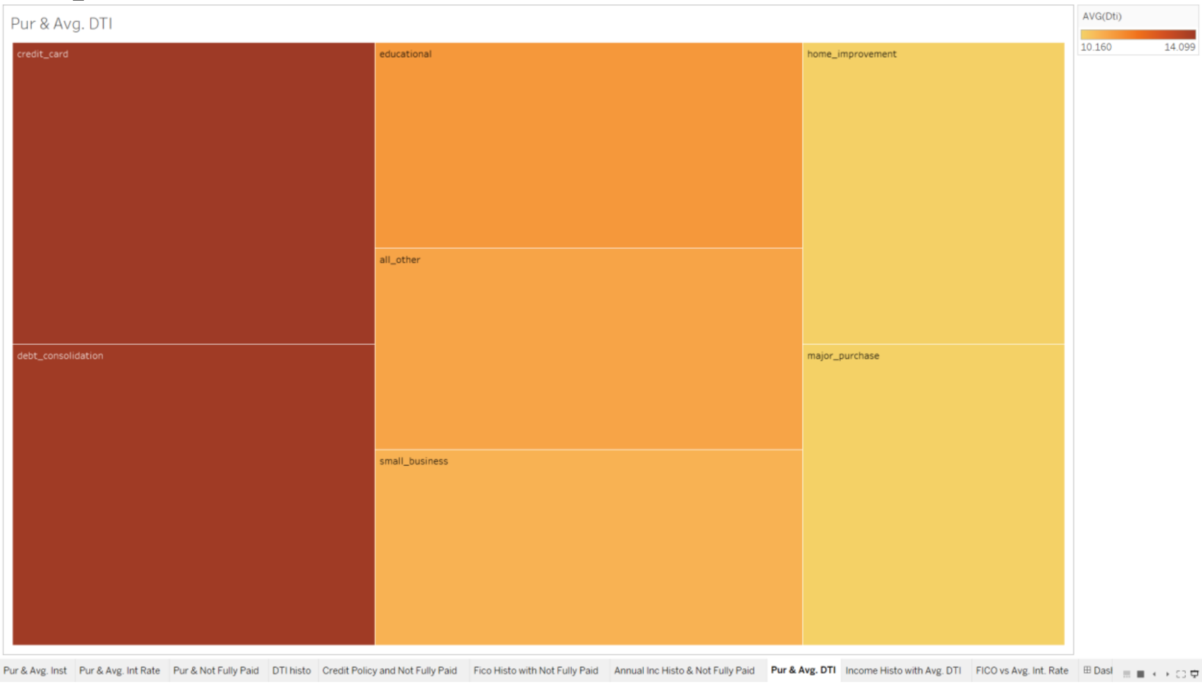
Graph 8:



Sheet 8: Divided Histogram of Annual Income

Most of the people have annual income between 10.2k – 11.3k and on an average 14.8% don't pay the full loan. Higher the annual income higher is the probability of the person paying the full loan and vise-versa.

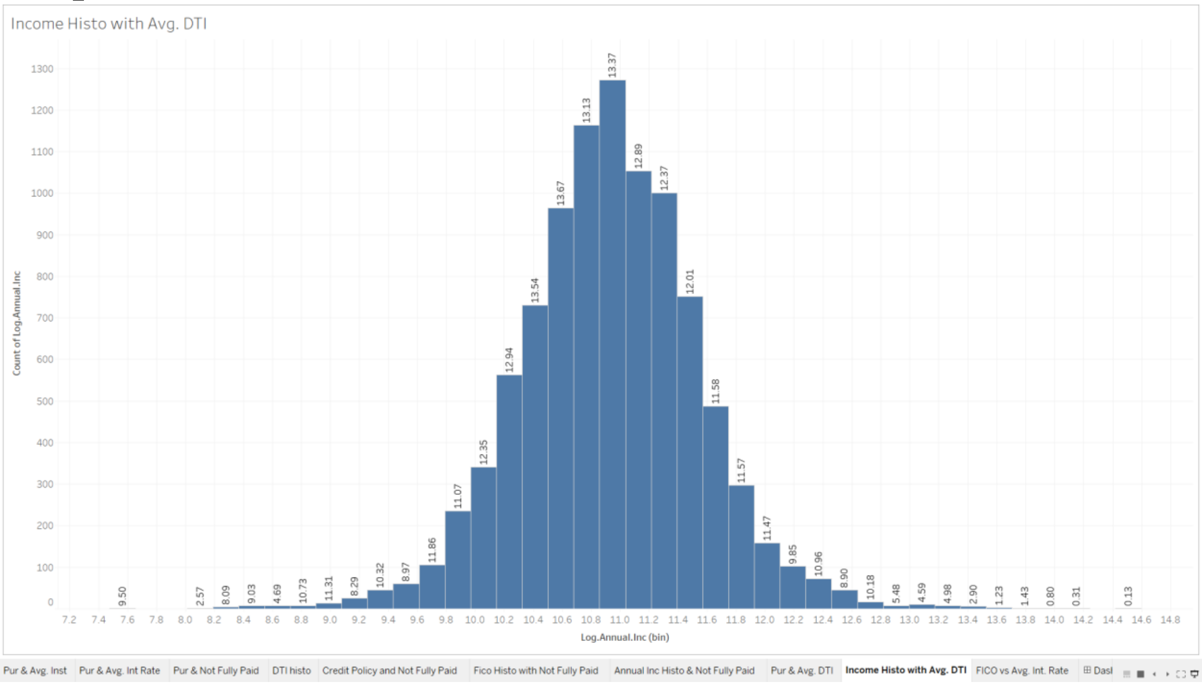
Graph 9:



Sheet 9: TreeMap of Purpose and Average DTI

Average DTI is highest for the people who take loan for credit card purpose i.e. 14.099% and lowest for major purchase purpose i.e. 10.16%. Also people who take loan for debt consolidation have high, home improvement have low and educational, small business and all other purposes have medium DTI.

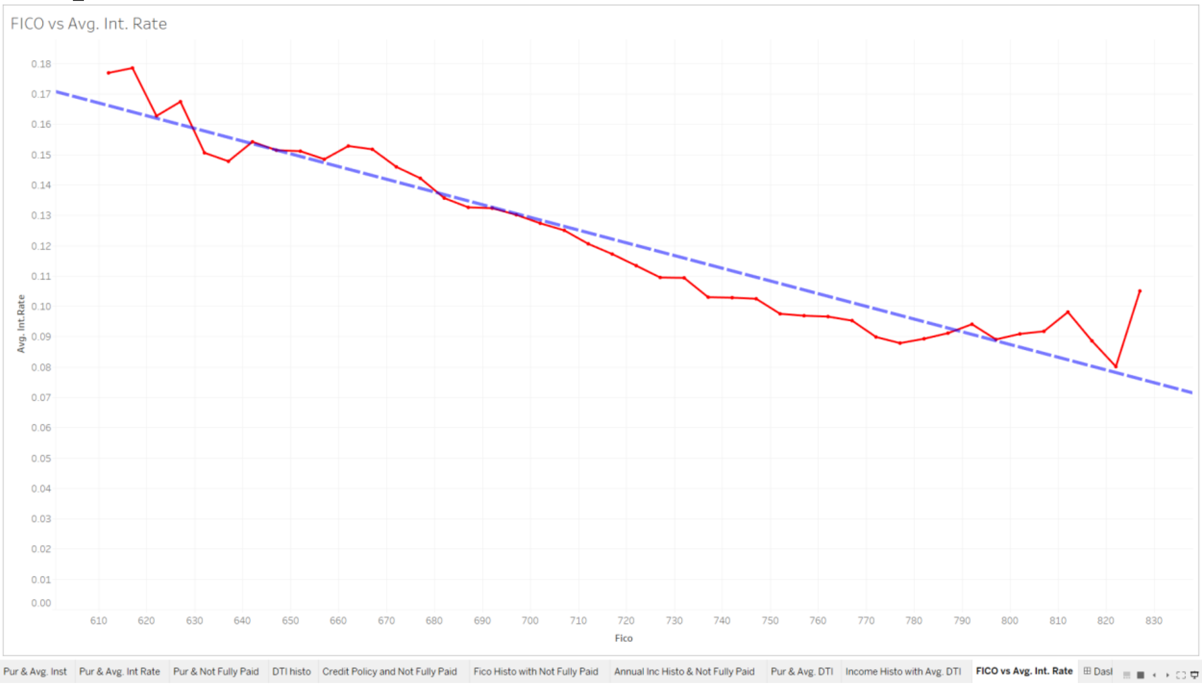
Graph 10:



Sheet 10: Histogram of Annual Income with Average DTI

Most of the people have annual income between 10.2k – 11.3k and high DTI (10-14) between this range. Higher the annual income lower is the DTI but vise versa is not true. People with high income spend less on debt clearance because they don’t need to take loan.

Graph 11:

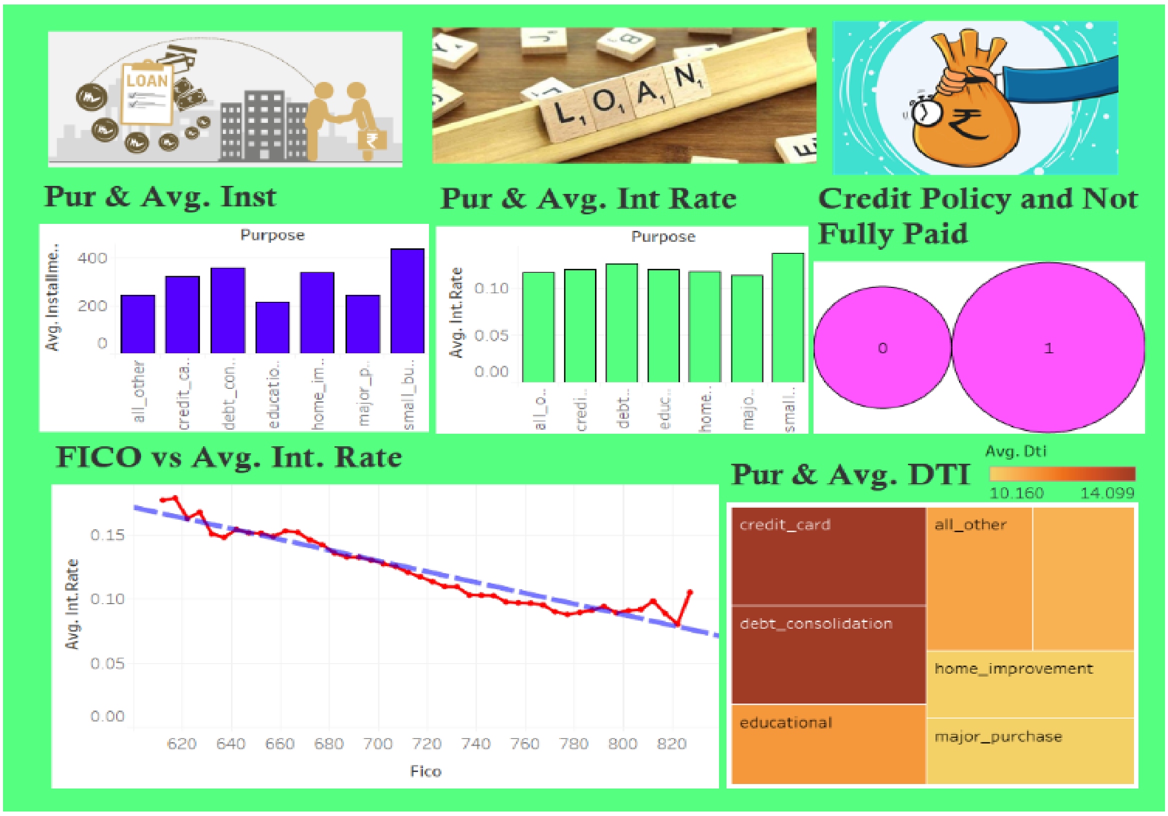


Sheet 11: Line Graph of FICO score and Average Interest Rate

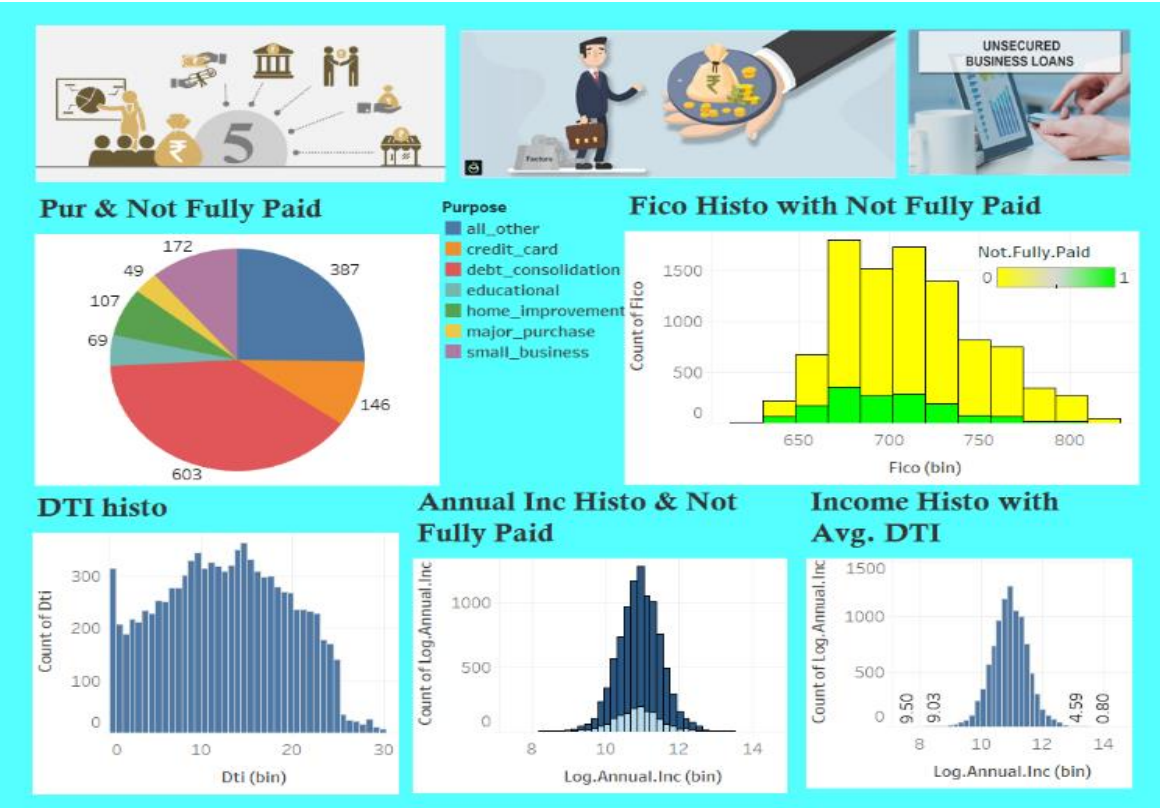
FICO score and Average Interest rate are inversely proportional to each other. As FICO score increases Avg. Interest Rate decreases. The relationship is linear with slope m = -0.000419076 and intercept c = 0.422622.

Avg. Int.Rate = -0.000419076*Fico + 0.422622

Dashboard 1:



Dashboard 2:



Inferences :

1. Highest avg. installment is done for small businesses and lowest is done for educational purpose
2. Most of the loans are taken for debt consolidation, about 3957 (41%). Least amount of loans are taken for Educational purpose, about 343 (3.5%).
3. Interest Rate is highest for small business purpose (0.138) and lowest for major purchase (0.114).
4. Most of the people (39.33%) who took loan for debt consolidation are not able to pay the full loan
5. Most of the people who have a credit policy are not able to pay the full loan (66.14%). The people who don’t have a credit policy and are not able to pay the full loan are 33.85%.
6. Most of the people have annual income between 10.2k – 11.3k and on an average 14.8% don’t pay the full loan.
7. FICO score and Average Interest rate are inversely proportional to each other. As FICO score increases Avg. Interest Rate decreases

PROJECT : 2

TOPIC : Fake News Classification

Table of Contents

Problem Statement(AIM):

To predict the news articles as Fake news or Real news.

Objectives:

1. Gathering and pre-processing the data as per needed.
2. Finding an appropriate model and training data on the same.
3. Testing the accuracy of model and predicting the result.

Theory:

Fake news

Fake news is false or misleading information presented as news. It often has the aim of damaging the reputation of a person or entity, or making money through advertising revenue.

tf–idf:

In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.[1] It is often used as a weighting factor in searches of information retrieval, text mining, and user modelling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. tf–idf is one of the most popular term-weighting schemes today. A survey conducted in 2015 showed that 83% of text-based recommender systems in digital libraries use tf–idf.[2]

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.

One of the simplest ranking functions is computed by summing the tf–idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Passive Aggressive Classifiers:

The Passive-Aggressive algorithms are a family of Machine learning algorithms that are not very well known by beginners and even intermediate Machine Learning enthusiasts. However, they can be very useful and efficient for certain applications.

Note: This is a high-level overview of the algorithm explaining how it works and when to use it. It does not go deep into the mathematics of how it works.

Passive-Aggressive algorithms are generally used for large-scale learning. It is one of the few ‘online-learning algorithms‘. In online machine learning algorithms, the input data comes in sequential order and the machine learning model is updated step-by-step, as opposed to batch learning, where the entire training dataset is used at once. This is very useful in situations where there is a huge amount of data and it is computationally infeasible to train the entire dataset because of the sheer size of the data. We can simply say that an online-learning algorithm will get a training example, update the classifier, and then throw away the example.

A very good example of this would be to detect fake news on a social media website like Twitter, where new data is being added every second. To dynamically read data from Twitter continuously, the data would be huge, and using an online-learning algorithm would be ideal.

Passive-Aggressive algorithms are somewhat similar to a Perceptron model, in the sense that they do not require a learning rate. However, they do include a regularization parameter.

How Passive-Aggressive Algorithms Work:

Passive-Aggressive algorithms are called so because:

Passive: If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model.

Aggressive: If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

Understanding the mathematics behind this algorithm is not very simple and is beyond the scope of a single article. This article provides just an overview of the algorithm and a simple implementation of it. To learn more about the mathematics behind this algorithm, I recommend watching this excellent video on the algorithm's working by Dr Victor Lavrenko.

Important parameters:

C: This is the regularization parameter, and denotes the penalization the model will make on an incorrect prediction

max_iter: The maximum number of iterations the model makes over the training data.

tol: The stopping criterion. If it is set to None, the model will stop when (loss > previous_loss – tol). By default, it is set to 1e-3.

Dataset:

The dataset contains various news articles based on title and text along with labels of if they are fake or real news.

It contains various features such text, title, labels. Has 40000 datapoints.

Out[2]:

	title	text	subject	date	label
0	PRESIDENT TRUMP Explains New “America First” R...	That s what we re talking about! Another campa...	politics	2-Aug-17	Fake
1	TERMINALLY ILL FORMER MISS WI: “Until my last ...	How is it that Sean Hannity is the only media ...	politics	4-Oct-16	Fake
2	Cruz Humiliated By Moderator After Lie About ...	Almost immediately after learning that longtim...	News	13-Feb-16	Fake
3	Russia revels in Trump victory, looks to sanct...	MOSCOW (Reuters) - For all their mutual praise...	politicsNews	9-Nov-16	Real
4	Trump's bid to open U.S. monuments to developm...	WASHINGTON (Reuters) - The Trump administratio...	politicsNews	26-May-17	Real

Code:

Importing the required libraries

In [1]:
```
import numpy as np
import pandas as pd
import itertools
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import joblib
```

Reading in the csv file

In [2]:
```
#Read the data
df=pd.read_csv('train.csv')

#Get shape and head
df.shape
df.head()
```

Out[2]:

	title	text	subject	date	label
0	PRESIDENT TRUMP Explains New “America First” R...	That s what we re talking about! Another campa...	politics	2-Aug-17	Fake
1	TERMINALLY ILL FORMER MISS WI: “Until my last ...	How is it that Sean Hannity is the only media ...	politics	4-Oct-16	Fake
2	Cruz Humiliated By Moderator After Lie About ...	Almost immediately after learning that longtim...	News	13-Feb-16	Fake
3	Russia revels in Trump victory, looks to sanct...	MOSCOW (Reuters) - For all their mutual praise...	politicsNews	9-Nov-16	Real

In [3]:
```
columnsList = df.columns
columnsList
```

Out[3]: `Index(['title', 'text', 'subject', 'date', 'label'], dtype='object')`

In [ ]:
```
df.count()
```

Finding out the null value count and which column they're in

In [4]:
```
df.isna().sum()
#as can be observed there are null values in the dataset and we can drop them since they arent too many
```

Out[4]:
```
title      38
text       38
subject    57
date       57
label      57
dtype: int64
```

In [ ]:
```
df.describe()
```

Dropping the rows with null values present since they arent too many

In [5]:
```
df = df.dropna()
df.count()
df.head(5)
```

Out[5]:

	title	text	subject	date	label
0	PRESIDENT TRUMP Explains New “America First” R...	That s what we re talking about! Another campa...	politics	2-Aug-17	Fake
1	TERMINALLY ILL FORMER MISS WI: “Until my last ...	How is it that Sean Hannity is the only media ...	politics	4-Oct-16	Fake
2	Cruz Humiliated By Moderator After Lie About ...	Almost immediately after learning that longtim...	News	13-Feb-16	Fake
3	Russia revels in Trump victory, looks to sanct...	MOSCOW (Reuters) - For all their mutual praise...	politicsNews	9-Nov-16	Real
4	Trump's bid to open U.S. monuments to developm...	WASHINGTON (Reuters) - The Trump administratio...	politicsNews	26-May-17	Real

```
In [ ]: df['label'].value_counts()
```

```
In [7]: #Get the labels
        labels=df.label
        labels.head()
```

```
Out[7]: 0    Fake
        1    Fake
        2    Fake
        3    Real
        4    Real
        Name: label, dtype: object
```

Splitting the data into train-test and then training and testing the model

```
In [8]: #Split the dataset
        x_train,x_test,y_train,y_test=train_test_split(df['title'], labels, test_size=0.2, random_state=7)
```

```
In [9]: #Initialize a TfidfVectorizer
        tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)

        #Fit and transform train set, transform test set
        tfidf_train=tfidf_vectorizer.fit_transform(x_train)
        tfidf_test=tfidf_vectorizer.transform(x_test)
```

```
In [10]: #Initialize a PassiveAggressiveClassifier
         pac=PassiveAggressiveClassifier(max_iter=50)
         pac.fit(tfidf_train,y_train)

         #Predict on the test set and calculate accuracy
         y_pred=pac.predict(tfidf_test)
         score=accuracy_score(y_test,y_pred)
         print(f'Accuracy: {round(score*100,2)}%')

         Accuracy: 93.64%
```

```
In [11]: #Build confusion matrix
         confusion_matrix(y_test,y_pred, labels=['Fake','Real'])

         #4000 Fake were correctly predicted, while 243 were not
         #255 Real were not correctly predicted while 3500 were

Out[11]: array([[4001,  242],
                [ 267, 3488]], dtype=int64)
```

```
In [12]: predHeadline = " Sarah Palin Just Openly Admitted She Prefers Being Interviewed By Children Over The Press"
         predVec = tfidf_vectorizer.transform([predHeadline])
         pred = pac.predict(predVec)
```


```
In [12]: predHeadline = " Sarah Palin Just Openly Admitted She Prefers Being Interviewed By Children Over The Press"
         predVec = tfidf_vectorizer.transform([predHeadline])
         pred = pac.predict(predVec)
```


```
In [13]: pred

Out[13]: array(['Fake'], dtype='<U414')
```

Output:

1. Confusion Matrix:

```
In [11]: #Build confusion matrix
         confusion_matrix(y_test,y_pred, labels=['Fake','Real'])

         #4001 Fake were correctly predicted, while 242 were not
         #267 Real were not correctly predicted while 3488 were

Out[11]: array([[4001,  242],
                [ 267, 3488]], dtype=int64)
```

2. Testing on Unlabelled Data:

In [16]:
```
predTitles = dtest['title']
predTitles
```

Out[16]:
```
0       STUNNING TESTIMONY On The Devastation Illegal ...
1       Pena Nieto told Trump Mexico won't pay for wal...
2        Actor George Clooney: I Support Hillary, And ...
3       AWESOME LETTER TO OBAMA: Who is unfit to be pr...
4        This Ridiculously Creepy Vine May Prove Ted C...
                              ...
3995    WHERE’S THE OVERSIGHT? OBAMA FUNNELED BILLIONS...
3996    Trump says he is 'very, very close' to making ...
3997     NSA Chief On Trump’s Russia/US Cybersecurity ...
3998     Chris Christie Says Only Stupid Judges Will W...
3999     The Internet EVISCERATES Whiny Nikki Haley Af...
Name: title, Length: 4000, dtype: object
```

In [17]:
```
predictionTestVector = tfidf_vectorizer.transform(predTitles)
predTest = pac.predict(predictionTestVector)
```

In [18]:
```
predTest
```

Out[18]:
```
array(['Fake', 'Real', 'Fake', ..., 'Real', 'Real', 'Fake'], dtype='<U414')
```

Inference:

1. The model has an accuracy of 93.64%.
2. With the above results it can be inferred that given sufficient data, we can train models that can classify news as Fake or not, given our current digital age this is a vital task to be carried out. To test this, the model was also run on an unlabelled dataset.

From the confusion matrix it can be observed that there are #4001 Fake were correctly predicted, while 242 were not, #267 Real were not correctly predicted while 3488 were.