In order to increase the bank profit and income,Thera bank wants to increase their borrowers rate.
And their number of depositors are quite large compare to borrowers.Moreover,they have a successful campaign history of rate 9% where 9% of depositors were interested in taking loan.This transition made bank to prepare for better campaign.

The file Bank.xls contains data on 5000 customers. The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

Let’s take the list of attributes which are considered for this campaign

Age
Experience
Income
ZIPCode
Family
CCAvg
Education
Mortgage
Personal Loan
Securities Account
CD Account
Online
CreditCard

Let us find dependent and independent variables for the output of chances of taking loan.

Obviously,Bank is interested in people who will be able to pay the loan back.For that,as a first step let us analyse the parameters which decides the eligible people.

Attributes	To be considered for analysis	Reason
Age	yes	Chances of taking loan of aged person is low but young people will be ready to take loan.ie..lesser the age higher the chance of getting loan
Experience	no	Age and experience are closely related
Income	yes	Higher income group will be attracted to the loan
ZIPCode	yes	Rural or urban.Chances of urban people taking loan would be high

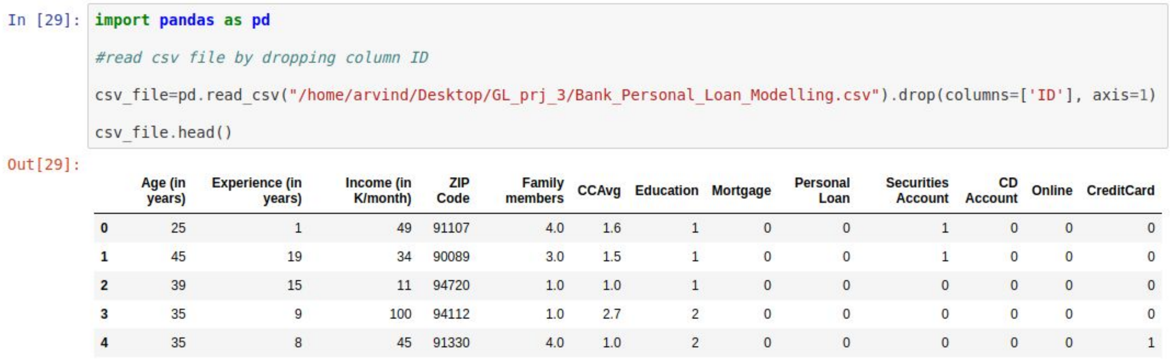
Family	maybe	Not necessary
CCAvg	no	Not necessary
Education	yes	Higher qualified people would be able to pay the loan back easily
Mortgage	yes	If the person has mortgage already then he is not likely to take loan
Personal Loan	yes	If the person has already taken personal loan then the chances are less
Securities Account	yes	Bank can give loan to person who has securities account with them already
CD Account	no	
Online	Maybe	
CreditCard	yes	This shows the commitment of person towards loan

Business Objective of Thera Bank:

The main business objective of Thera Bank campaign is to increase their income through personal loan interest.Moreover,the bank wants to balance the ratio between depositors and borrowers.

Exploratory Data Analysis

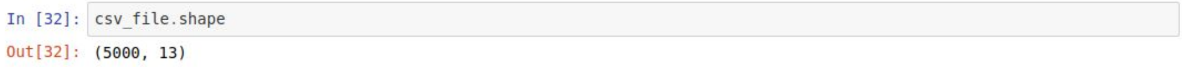
- As a first step,import input file as csv and get the head of file,which displays first 5 rows.
- From the display of head it is clear that the file has 14 rows of numerical value(non string) values.
- Where as,Column ‘Family members’ and ‘CCAvg’ has float values.



```
In [29]: import pandas as pd

         #read csv file by dropping column ID

         csv_file=pd.read_csv("/home/arvind/Desktop/GL_prj_3/Bank_Personal_Loan_Modelling.csv").drop(columns=['ID'], axis=1)

         csv_file.head()
Out[29]:
```

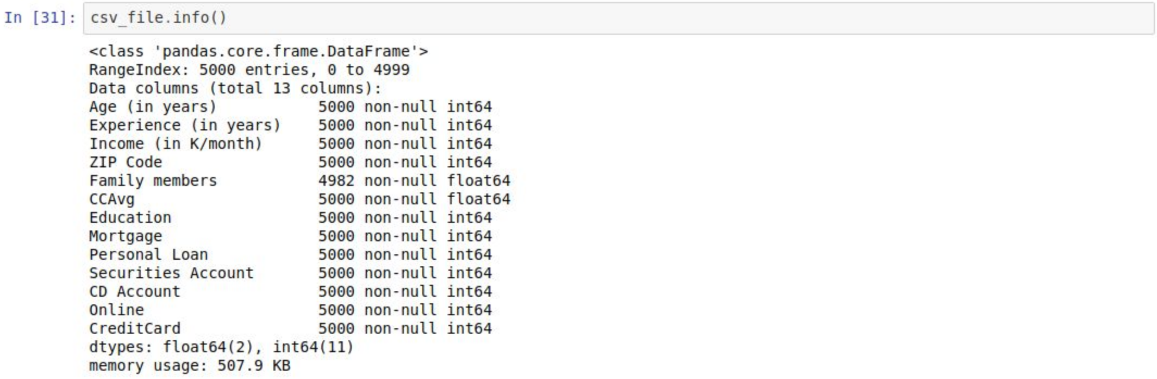
	Age (in years)	Experience (in years)	Income (in K/month)	ZIP Code	Family members	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	25	1	49	91107	4.0	1.6	1	0	0	1	0	0	0
1	45	19	34	90089	3.0	1.5	1	0	0	1	0	0	0
2	39	15	11	94720	1.0	1.0	1	0	0	0	0	0	0
3	35	9	100	94112	1.0	2.7	2	0	0	0	0	0	0
4	35	8	45	91330	4.0	1.0	2	0	0	0	0	0	1

- shape() tells us the size of file.(5000,13) which means 5000 rows and 13 columns.



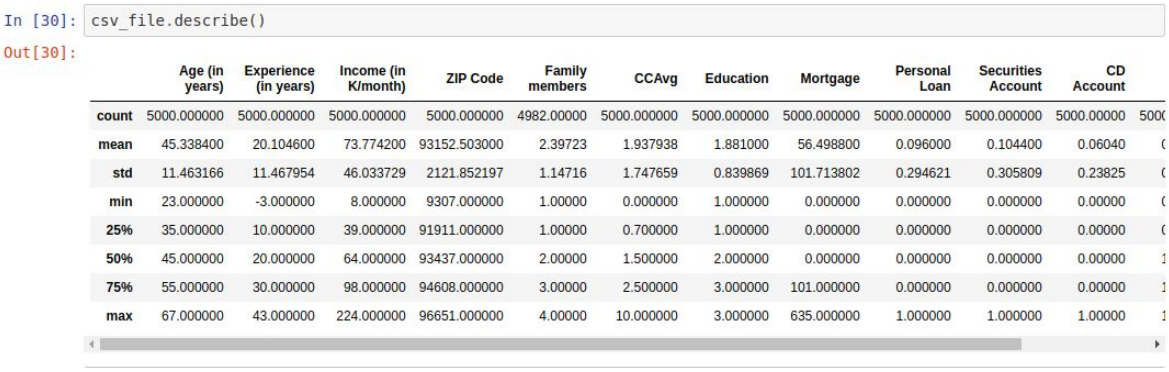
```
In [32]: csv_file.shape
Out[32]: (5000, 13)
```

- From info() ,we can infer the information such as,
    - Null values.Here ‘Family members’ has only 4982 entries.which means it has 18 null entries.
    - Except ‘Family members’ and ‘CCAvg’ (which are float64) all other attributes are int64



```
In [31]: csv_file.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
Age (in years)           5000 non-null int64
Experience (in years)    5000 non-null int64
Income (in K/month)      5000 non-null int64
ZIP Code                 5000 non-null int64
Family members           4982 non-null float64
CCAvg                    5000 non-null float64
Education                5000 non-null int64
Mortgage                 5000 non-null int64
Personal Loan            5000 non-null int64
Securities Account       5000 non-null int64
CD Account               5000 non-null int64
Online                   5000 non-null int64
CreditCard               5000 non-null int64
dtypes: float64(2), int64(11)
memory usage: 507.9 KB
```

- describe() gives us the complete data summary such as mean,median,min,max etc..
- From this summary,it is very clear that Mortgage has outlier.

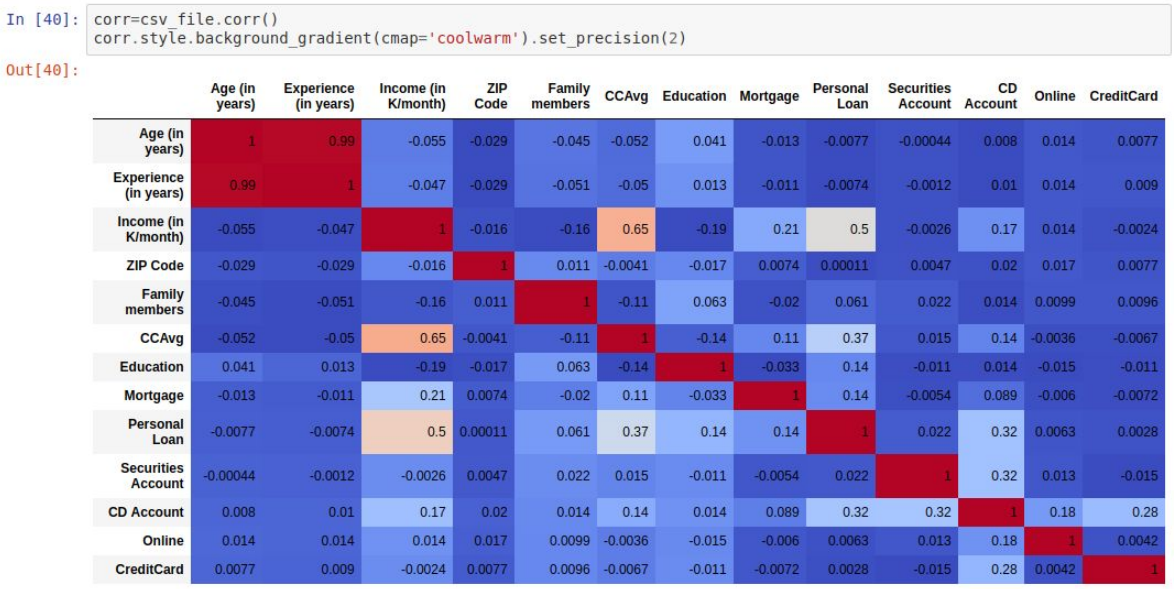


```
In [30]: csv_file.describe()
```

Out[30]:

	Age (in years)	Experience (in years)	Income (in K/month)	ZIP Code	Family members	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	
count	5000.000000	5000.000000	5000.000000	5000.000000	4982.00000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000	500
mean	45.338400	20.104600	73.774200	93152.503000	2.39723	1.937938	1.881000	56.498800	0.096000	0.104400	0.06040	0
std	11.463166	11.467954	46.033729	2121.852197	1.14716	1.747659	0.839869	101.713802	0.294621	0.305809	0.23825	0
min	23.000000	-3.000000	8.000000	9307.000000	1.00000	0.000000	1.000000	0.000000	0.000000	0.000000	0.00000	0
25%	35.000000	10.000000	39.000000	91911.000000	1.00000	0.700000	1.000000	0.000000	0.000000	0.000000	0.00000	0
50%	45.000000	20.000000	64.000000	93437.000000	2.00000	1.500000	2.000000	0.000000	0.000000	0.000000	0.00000	1
75%	55.000000	30.000000	98.000000	94608.000000	3.00000	2.500000	3.000000	101.000000	0.000000	0.000000	0.00000	1
max	67.000000	43.000000	224.000000	96651.000000	4.00000	10.000000	3.000000	635.000000	1.000000	1.000000	1.00000	1

- Here darker shades indicated positive correlation and lighter shades indicates negative correlation.
- Age and Experience is highly correlated.
- Similarly,income and CCAvg ,income and personal loan is moderately correlated.
- Zip code,credit card,online not correlated to any of the attributes.



```
corr=csv_file.corr()
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

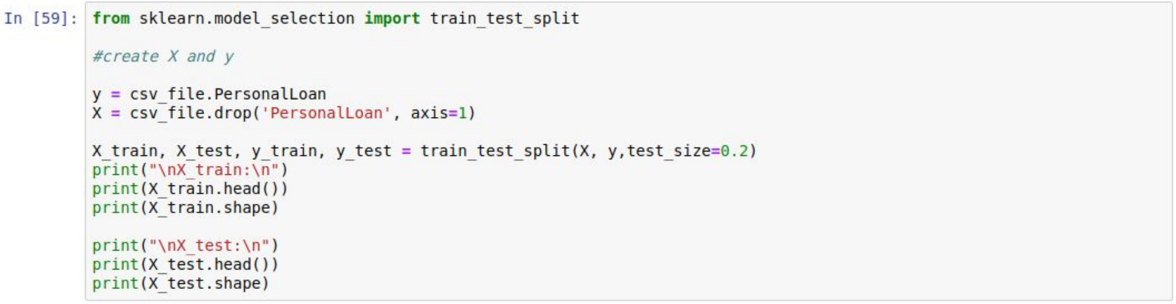
	Age (in years)	Experience (in years)	Income (in K/month)	ZIP Code	Family members	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
Age (in years)	1	0.99	-0.055	-0.029	-0.045	-0.052	0.041	-0.013	-0.0077	-0.00044	0.008	0.014	0.0077
Experience (in years)	0.99	1	-0.047	-0.029	-0.051	-0.05	0.013	-0.011	-0.0074	-0.0012	0.01	0.014	0.009
Income (in K/month)	-0.055	-0.047	1	-0.016	-0.16	0.65	-0.19	0.21	0.5	-0.0026	0.17	0.014	-0.0024
ZIP Code	-0.029	-0.029	-0.016	1	0.011	-0.0041	-0.017	0.0074	0.00011	0.0047	0.02	0.017	0.0077
Family members	-0.045	-0.051	-0.16	0.011	1	-0.11	0.063	-0.02	0.061	0.022	0.014	0.0099	0.0096
CCAvg	-0.052	-0.05	0.65	-0.0041	-0.11	1	-0.14	0.11	0.37	0.015	0.14	-0.0036	-0.0067
Education	0.041	0.013	-0.19	-0.017	0.063	-0.14	1	-0.033	0.14	-0.011	0.014	-0.015	-0.011
Mortgage	-0.013	-0.011	0.21	0.0074	-0.02	0.11	-0.033	1	0.14	-0.0054	0.089	-0.006	-0.0072
Personal Loan	-0.0077	-0.0074	0.5	0.00011	0.061	0.37	0.14	0.14	1	0.022	0.32	0.0063	0.0028
Securities Account	-0.00044	-0.0012	-0.0026	0.0047	0.022	0.015	-0.011	-0.0054	0.022	1	0.32	0.013	-0.015
CD Account	0.008	0.01	0.17	0.02	0.014	0.14	0.014	0.089	0.32	0.32	1	0.18	0.28
Online	0.014	0.014	0.014	0.017	0.0099	-0.0036	-0.015	-0.006	0.0063	0.013	0.18	1	0.0042
CreditCard	0.0077	0.009	-0.0024	0.0077	0.0096	-0.0067	-0.011	-0.0072	0.0028	-0.015	0.28	0.0042	1

Hypothesis Statement

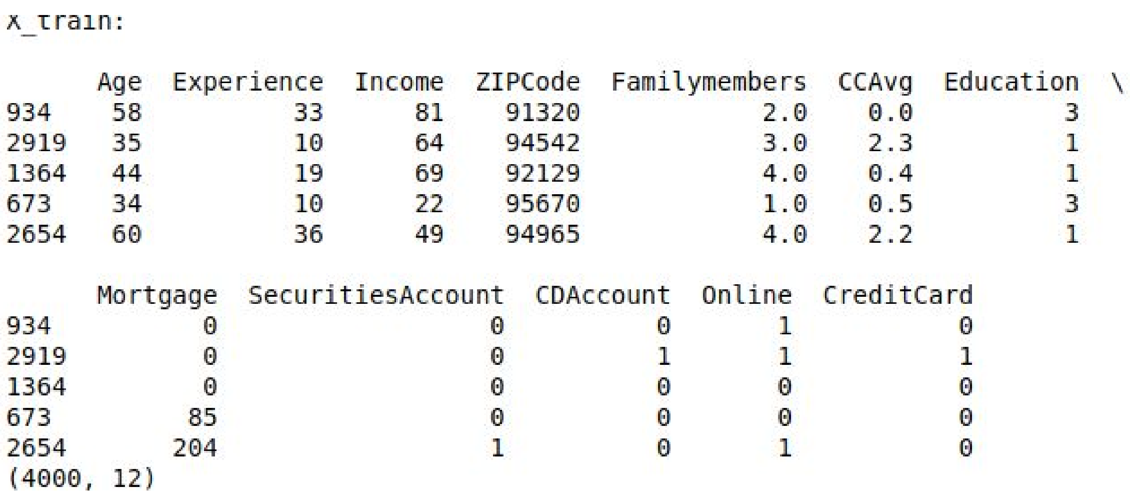
Hypothesis Validation

Splitting data in train and test dataset

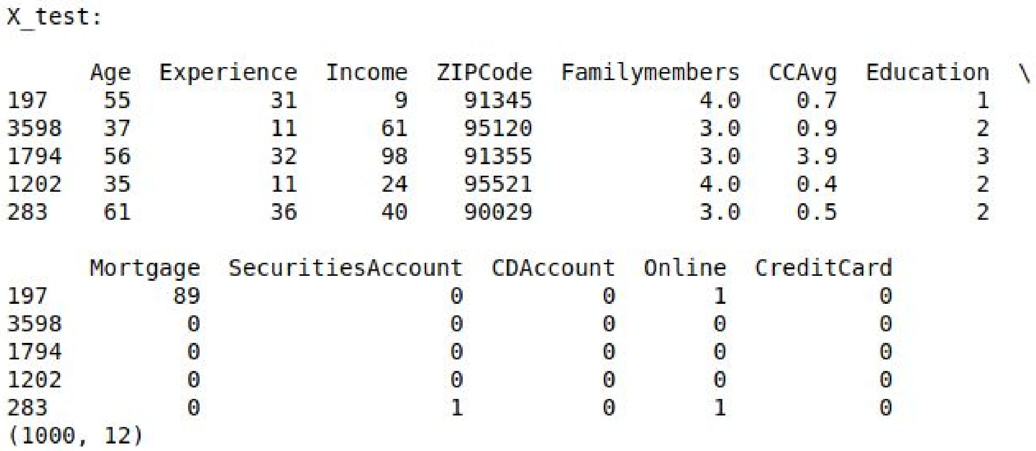
- Considering personal loan as dependent variable(Y) and rest of the variables as independent variable(X).



```
from sklearn.model_selection import train_test_split

#create X and y

y = csv_file.PersonalLoan
X = csv_file.drop('PersonalLoan', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2)
print("\nX_train:\n")
print(X_train.head())
print(X_train.shape)

print("\nX_test:\n")
print(X_test.head())
print(X_test.shape)
```



```
X_train:

      Age  Experience  Income  ZIPCode  Familymembers  CCAvg  Education  \
934    58          33      81    91320            2.0    0.0          3
2919   35          10      64    94542            3.0    2.3          1
1364   44          19      69    92129            4.0    0.4          1
673    34          10      22    95670            1.0    0.5          3
2654   60          36      49    94965            4.0    2.2          1

      Mortgage  SecuritiesAccount  CDAccount  Online  CreditCard
934          0                  0          0       1           0
2919         0                  0          1       1           1
1364         0                  0          0       0           0
673         85                  0          0       0           0
2654       204                  1          0       1           0
(4000, 12)
```



```
X_test:

      Age  Experience  Income  ZIPCode  Familymembers  CCAvg  Education  \
197    55          31       9    91345            4.0    0.7          1
3598   37          11      61    95120            3.0    0.9          2
1794   56          32      98    91355            3.0    3.9          3
1202   35          11      24    95521            4.0    0.4          2
283    61          36      40    90029            3.0    0.5          2

      Mortgage  SecuritiesAccount  CDAccount  Online  CreditCard
197         89                  0          0       1           0
3598         0                  0          0       0           0
1794         0                  0          0       0           0
1202         0                  0          0       0           0
283          0                  1          0       1           0
(1000, 12)
```

Feature engineering

- Creating new features with the help of existing attributes is called feature engineering.

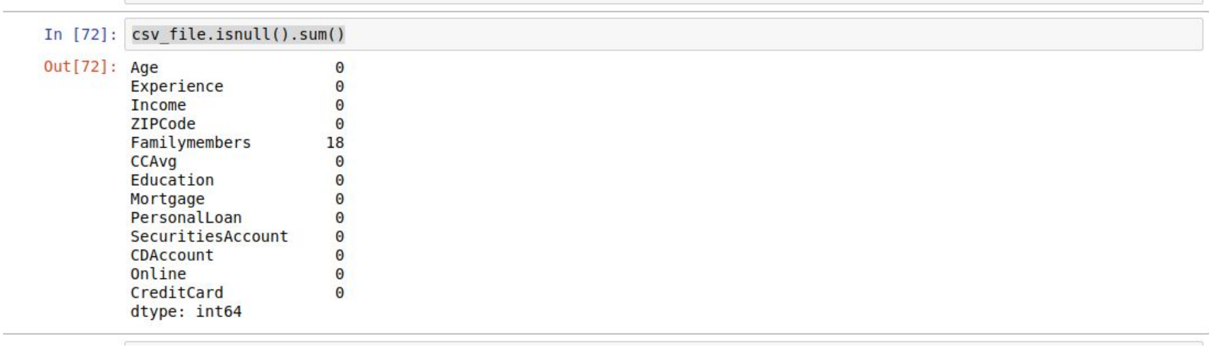
Here are the existing features,

ID	Customer ID
Age	Customer's age in years
Experience	Years of professional experience
Income	Annual income of the customer ($000)

ZIPCode	Home Address ZIP code.
Family	Family size of the customer
CCAvg	Avg. spending on credit cards per month ($000)
Education	Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional
Mortgage	Value of house mortgage if any. ($000)
Personal Loan	Did this customer accept the personal loan offered in the last campaign?
Securities Account	Does the customer have a securities account with the bank?
CD Account	Does the customer have a certificate of deposit (CD) account with the bank?
Online	Does the customer use internet banking facilities?
CreditCard	Does the customer use a credit card issued by the bank?

Before going more deep into feature engineering,let’s find the missing values.

- Familymembers has 18 missing values



```
In [72]: csv_file.isnull().sum()

Out[72]: Age                    0
         Experience             0
         Income                 0
         ZIPCode                0
         Familymembers         18
         CCAvg                  0
         Education              0
         Mortgage               0
         PersonalLoan           0
         SecuritiesAccount      0
         CDAccount              0
         Online                 0
         CreditCard             0
         dtype: int64
```

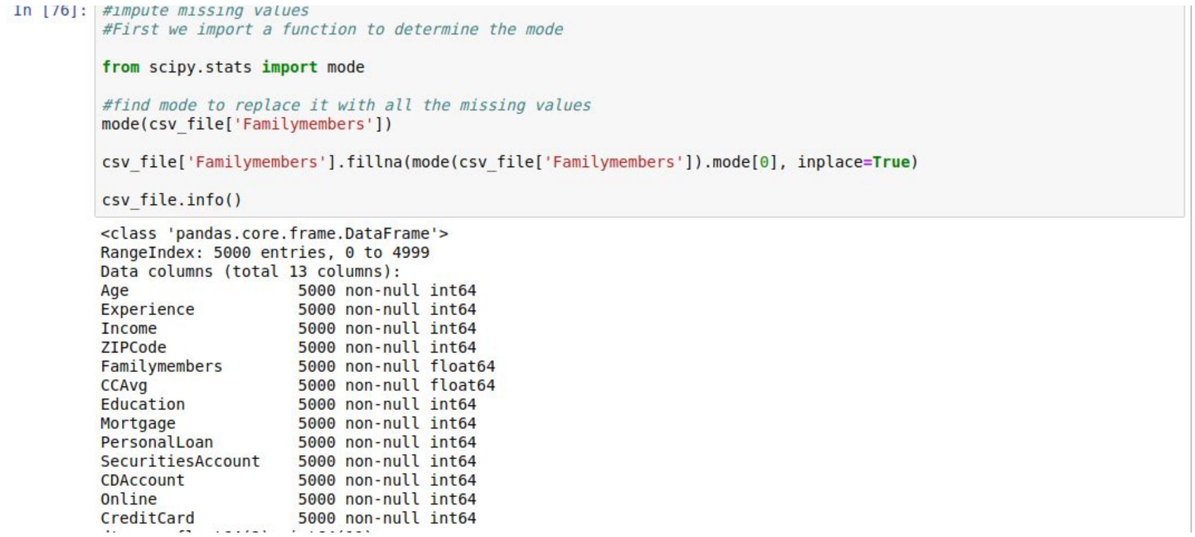
Feature engineering means building additional features out of existing data which is often spread across multiple related tables.

Featuretools is an open source library for performing automated feature engineering.

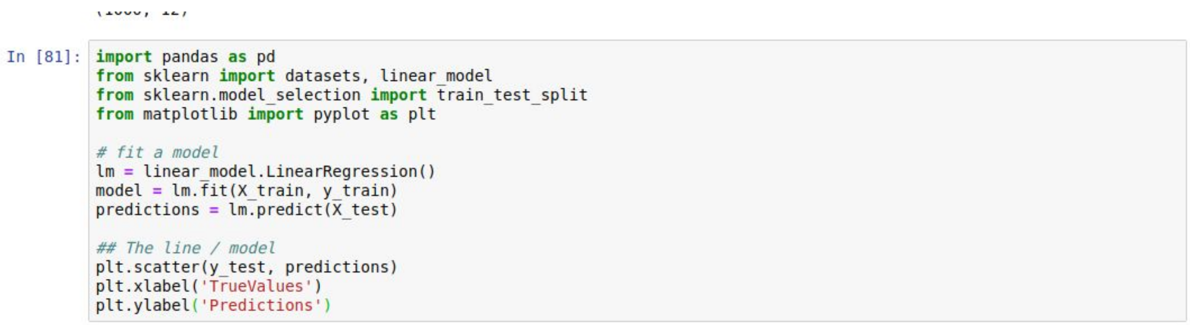
The first two concepts of featuretools are entities and entitysets. An entity is simply a table and an EntitySet is a collection of tables and the relationships between them.

Charts and Graphs to show the relationship between independent and dependent variables

- Impute the missing values before plotting the graph.

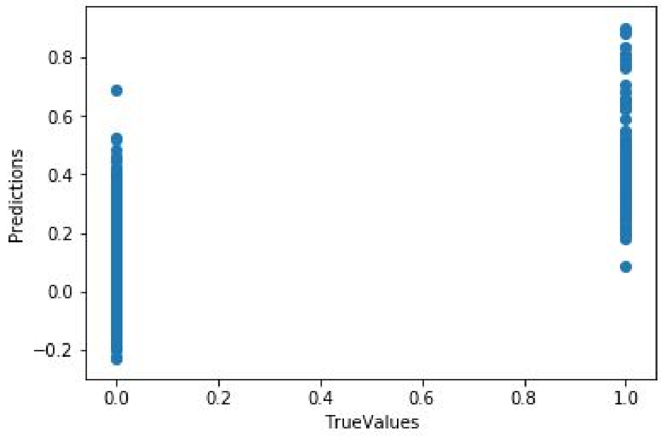


```
In [76]: #impute missing values
         #First we import a function to determine the mode

         from scipy.stats import mode

         #find mode to replace it with all the missing values
         mode(csv_file['Familymembers'])

         csv_file['Familymembers'].fillna(mode(csv_file['Familymembers']).mode[0], inplace=True)

         csv_file.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 5000 entries, 0 to 4999
         Data columns (total 13 columns):
         Age                  5000 non-null int64
         Experience           5000 non-null int64
         Income               5000 non-null int64
         ZIPCode              5000 non-null int64
         Familymembers        5000 non-null float64
         CCAvg                5000 non-null float64
         Education            5000 non-null int64
         Mortgage             5000 non-null int64
         PersonalLoan         5000 non-null int64
         SecuritiesAccount    5000 non-null int64
         CDAccount            5000 non-null int64
         Online               5000 non-null int64
         CreditCard           5000 non-null int64
```



```
         (1000, 12)

In [81]: import pandas as pd
         from sklearn import datasets, linear_model
         from sklearn.model_selection import train_test_split
         from matplotlib import pyplot as plt

         # fit a model
         lm = linear_model.LinearRegression()
         model = lm.fit(X_train, y_train)
         predictions = lm.predict(X_test)

         ## The line / model
         plt.scatter(y_test, predictions)
         plt.xlabel('TrueValues')
         plt.ylabel('Predictions')
```

Out[81]: Text(0, 0.5, 'Predictions')



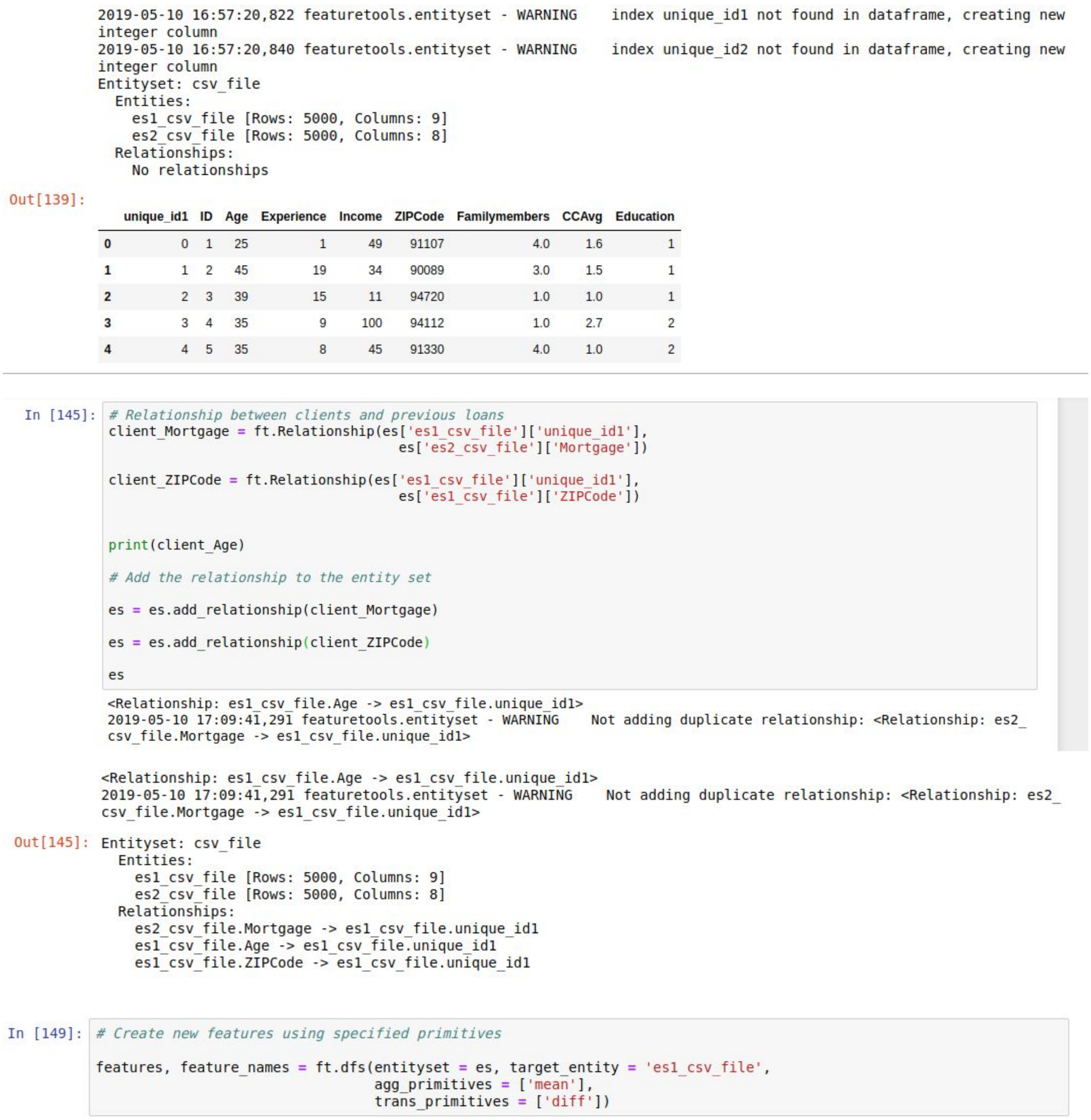
In [139]:

```
import featuretools as ft

#i have splitted the csv file in order to get two entities.

es1_csv_file=pd.read_csv("/home/arvind/Desktop/GL_prj_3/Bank_Personal_Loan_Modelling_es1.csv")

es2_csv_file=pd.read_csv("/home/arvind/Desktop/GL_prj_3/Bank_Personal_Loan_Modelling_es2.csv")

# Create new entityset

es = ft.EntitySet(id = 'csv_file')

# Create an entity

# This dataframe already has an index and a time index

es = es.entity_from_dataframe(entity_id = 'es1_csv_file', dataframe = es1_csv_file,
                              index = 'unique_id1')

es = es.entity_from_dataframe(entity_id = 'es2_csv_file', dataframe = es2_csv_file,
                              index = 'unique_id2')
print(es)

es1_csv_file.head()
```

```
2019-05-10 16:57:20,822 featuretools.entityset - WARNING    index unique_id1 not found in dataframe, creating new
integer column
2019-05-10 16:57:20,840 featuretools.entityset - WARNING    index unique_id2 not found in dataframe, creating new
integer column
Entityset: csv_file
  Entities:
    es1_csv_file [Rows: 5000, Columns: 9]
    es2_csv_file [Rows: 5000, Columns: 8]
  Relationships:
    No relationships
```

Out[139]:

	unique_id1	ID	Age	Experience	Income	ZIPCode	Familymembers	CCAvg	Education
0	0	1	25	1	49	91107	4.0	1.6	1
1	1	2	45	19	34	90089	3.0	1.5	1
2	2	3	39	15	11	94720	1.0	1.0	1
3	3	4	35	9	100	94112	1.0	2.7	2
4	4	5	35	8	45	91330	4.0	1.0	2



```
2019-05-10 16:57:20,822 featuretools.entityset - WARNING    index unique_id1 not found in dataframe, creating new
integer column
2019-05-10 16:57:20,840 featuretools.entityset - WARNING    index unique_id2 not found in dataframe, creating new
integer column
Entityset: csv_file
  Entities:
    es1_csv_file [Rows: 5000, Columns: 9]
    es2_csv_file [Rows: 5000, Columns: 8]
  Relationships:
    No relationships
```

Out[139]:

	unique_id1	ID	Age	Experience	Income	ZIPCode	Familymembers	CCAvg	Education
0	0	1	25	1	49	91107	4.0	1.6	1
1	1	2	45	19	34	90089	3.0	1.5	1
2	2	3	39	15	11	94720	1.0	1.0	1
3	3	4	35	9	100	94112	1.0	2.7	2
4	4	5	35	8	45	91330	4.0	1.0	2

In [145]:

```
# Relationship between clients and previous loans
client_Mortgage = ft.Relationship(es['es1_csv_file']['unique_id1'],
                                  es['es2_csv_file']['Mortgage'])

client_ZIPCode = ft.Relationship(es['es1_csv_file']['unique_id1'],
                                  es['es1_csv_file']['ZIPCode'])


print(client_Age)

# Add the relationship to the entity set

es = es.add_relationship(client_Mortgage)

es = es.add_relationship(client_ZIPCode)

es
```

```
<Relationship: es1_csv_file.Age -> es1_csv_file.unique_id1>
2019-05-10 17:09:41,291 featuretools.entityset - WARNING    Not adding duplicate relationship: <Relationship: es2_
csv_file.Mortgage -> es1_csv_file.unique_id1>
```

```
<Relationship: es1_csv_file.Age -> es1_csv_file.unique_id1>
2019-05-10 17:09:41,291 featuretools.entityset - WARNING    Not adding duplicate relationship: <Relationship: es2_
csv_file.Mortgage -> es1_csv_file.unique_id1>
```

Out[145]:

```
Entityset: csv_file
  Entities:
    es1_csv_file [Rows: 5000, Columns: 9]
    es2_csv_file [Rows: 5000, Columns: 8]
  Relationships:
    es2_csv_file.Mortgage -> es1_csv_file.unique_id1
    es1_csv_file.Age -> es1_csv_file.unique_id1
    es1_csv_file.ZIPCode -> es1_csv_file.unique_id1
```

In [149]:

```
# Create new features using specified primitives

features, feature_names = ft.dfs(entityset = es, target_entity = 'es1_csv_file',
                                 agg_primitives = ['mean'],
                                 trans_primitives = ['diff'])
```

CART

Classification and Regression Trees or CART for short is an acronym introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification or regression predictive modeling problems.

Here the important part is gini index which helps s to decide the root node.

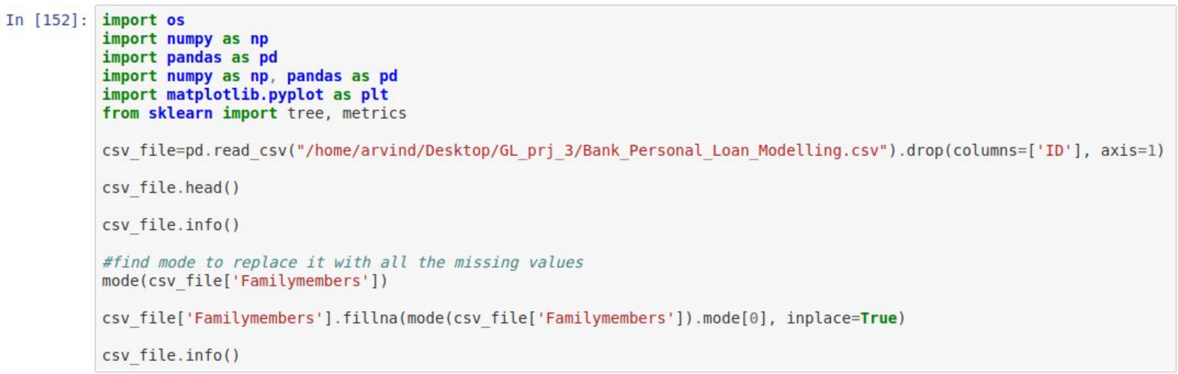
Gini Index

Gini index says, if we randomly select two items from a population, they must be of the same class and probability for this is 1 if the population is pure.

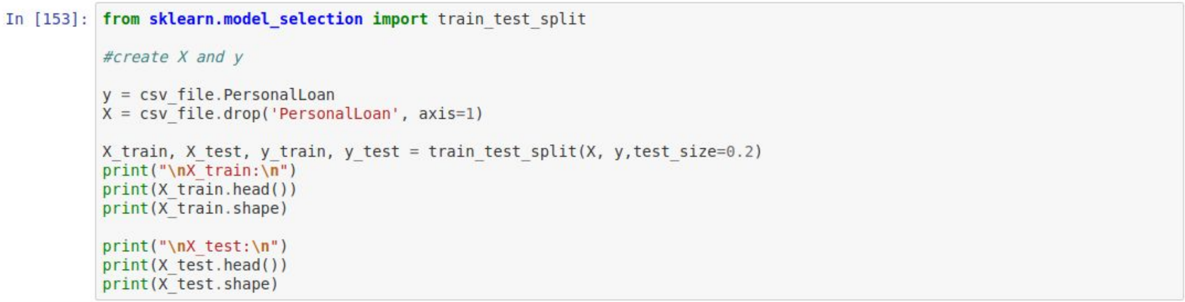
Higher the value of Gini, higher the homogeneity. CART (Classification and Regression Tree) uses the Gini method to create binary splits.

Steps to Calculate Gini for a split

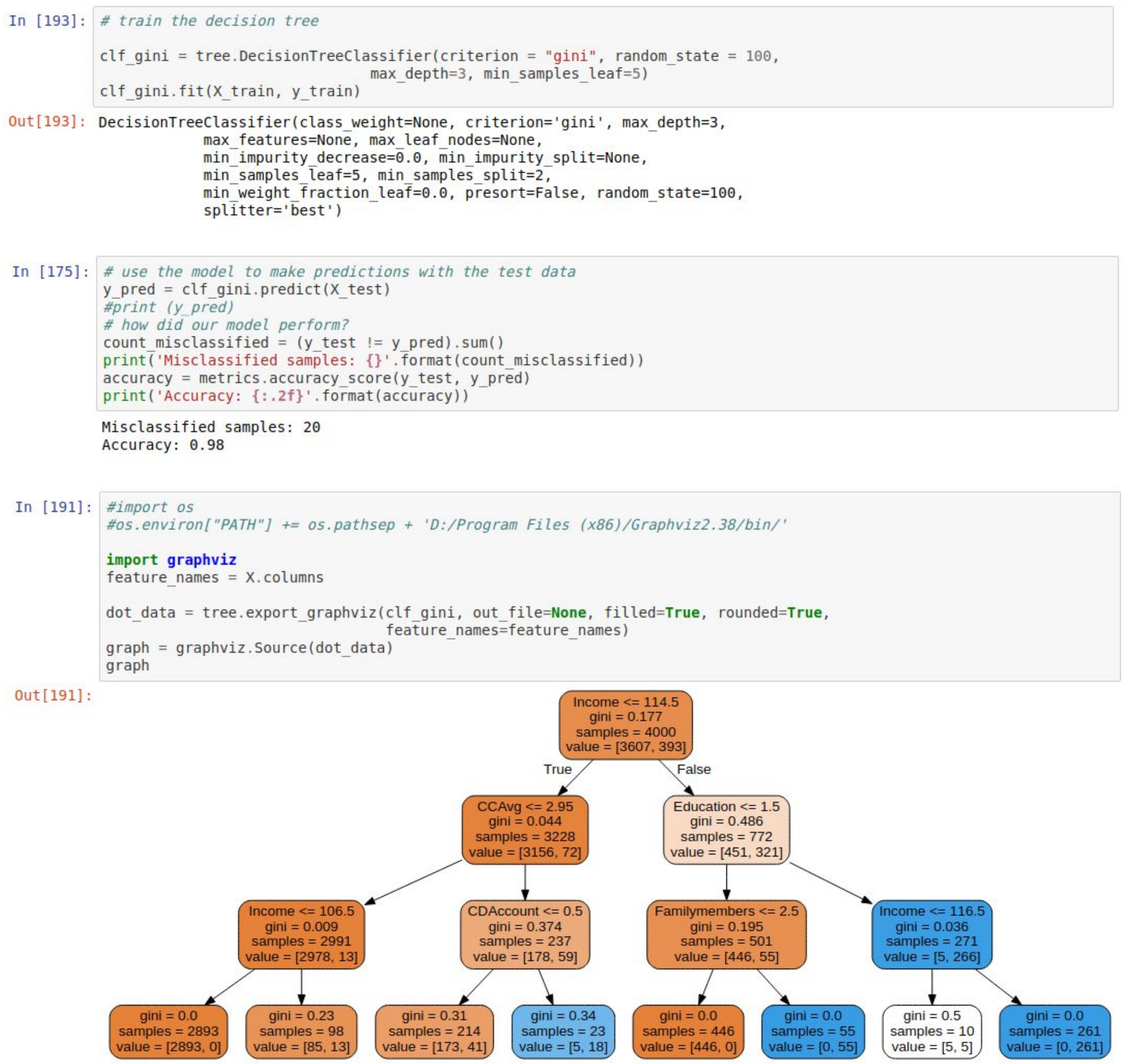
1. Calculate Gini for sub-nodes, using formula sum of the square of probability for success and failure (p^2+q^2).
2. Calculate Gini for split using weighted Gini score of each node of that split.



```
In [152]: import os
          import numpy as np
          import pandas as pd
          import numpy as np, pandas as pd
          import matplotlib.pyplot as plt
          from sklearn import tree, metrics

          csv_file=pd.read_csv("/home/arvind/Desktop/GL_prj_3/Bank_Personal_Loan_Modelling.csv").drop(columns=['ID'], axis=1)

          csv_file.head()

          csv_file.info()

          #find mode to replace it with all the missing values
          mode(csv_file['Familymembers'])

          csv_file['Familymembers'].fillna(mode(csv_file['Familymembers']).mode[0], inplace=True)

          csv_file.info()
```



```
In [153]: from sklearn.model_selection import train_test_split

          #create X and y

          y = csv_file.PersonalLoan
          X = csv_file.drop('PersonalLoan', axis=1)

          X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2)
          print("\nX_train:\n")
          print(X_train.head())
          print(X_train.shape)

          print("\nX_test:\n")
          print(X_test.head())
          print(X_test.shape)
```



```
In [193]: # train the decision tree

          clf_gini = tree.DecisionTreeClassifier(criterion = "gini", random_state = 100,
                                          max_depth=3, min_samples_leaf=5)
          clf_gini.fit(X_train, y_train)

Out[193]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=5, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort=False, random_state=100,
                      splitter='best')
```

```
In [175]: # use the model to make predictions with the test data
          y_pred = clf_gini.predict(X_test)
          #print (y_pred)
          # how did our model perform?
          count_misclassified = (y_test != y_pred).sum()
          print('Misclassified samples: {}'.format(count_misclassified))
          accuracy = metrics.accuracy_score(y_test, y_pred)
          print('Accuracy: {:.2f}'.format(accuracy))

          Misclassified samples: 20
          Accuracy: 0.98
```

```
In [191]: #import os
          #os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'

          import graphviz
          feature_names = X.columns

          dot_data = tree.export_graphviz(clf_gini, out_file=None, filled=True, rounded=True,
                                          feature_names=feature_names)
          graph = graphviz.Source(dot_data)
          graph

Out[191]:
```

Model performance measures

There are different kinds of metrics to evaluate our models. The choice of metric completely depends on the type of model and the implementation plan of the model. After you are finished building your model, these 7 metrics will help you in evaluating your model accuracy.

1. Confusion Matrix
2. Gain and Lift Chart
3. Kolmogorov Smirnov Chart
4. AUC – ROC
5. Gini Coefficient
6. Concordant – Discordant Ratio
7. Root Mean Squared Error
8. Cross Validation

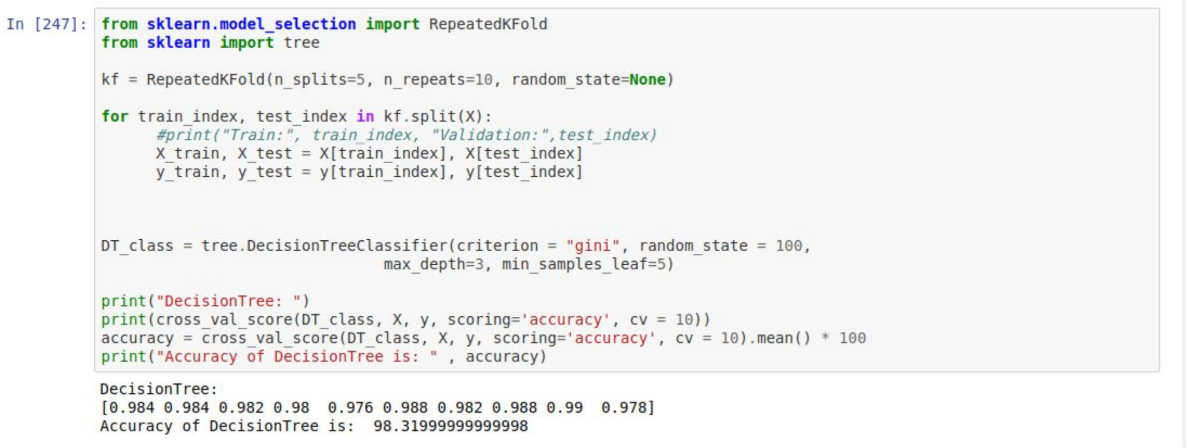
Model validation

K-fold cross validation is quite famous among the model validation measures.Because,

1. It trains the model on a large portion of the dataset.
2. It takes good ratio of testing data points. Less amount of data points can lead to a variance error while testing the effectiveness of the model
3. It iterates on the training and testing process multiple times.This helps in validating the model effectiveness properly

Below are the steps for it:

1. Randomly split your entire dataset into k”folds”
2. For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for *kth* fold
3. Record the error you see on each of the predictions
4. Repeat this until each of the k-folds has served as the test set
5. The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model



```
In [247]: from sklearn.model_selection import RepeatedKFold
          from sklearn import tree

          kf = RepeatedKFold(n_splits=5, n_repeats=10, random_state=None)

          for train_index, test_index in kf.split(X):
                #print("Train:", train_index, "Validation:",test_index)
                X_train, X_test = X[train_index], X[test_index]
                y_train, y_test = y[train_index], y[test_index]



          DT_class = tree.DecisionTreeClassifier(criterion = "gini", random_state = 100,
                                         max_depth=3, min_samples_leaf=5)

          print("DecisionTree: ")
          print(cross_val_score(DT_class, X, y, scoring='accuracy', cv = 10))
          accuracy = cross_val_score(DT_class, X, y, scoring='accuracy', cv = 10).mean() * 100
          print("Accuracy of DecisionTree is: " , accuracy)
```

```
DecisionTree:
[0.984 0.984 0.982 0.98  0.976 0.988 0.982 0.988 0.99  0.978]
Accuracy of DecisionTree is:  98.31999999999998
```

K-fold validation techniqe with 10 fold has validated the classifier with Accuracy of 98.32%