**CS 634 Data Mining Midterm Project**

**Name: Nehali Parulekar**

**Date: 28/04/2021**

**GitHub:** <https://github.com/NehaliParulekar/CS634Final_Project>

**Py Apriori Module:**

This is a basic implementation in Python of the Supervised Data Mining (Classification) Algorithms i.e. Naive Bayes Algorithm, Logistic Regression Algorithm, and Random Forest Algorithm.

**Requirements:**

**Software:** Python 3.7

Jupyter Notebook

**Libraries:** Pandas library is used to import the CSV file.

Itertools used to iterate over data structures that can be stepped over using a for-loop

NumPy used for working with arrays.

matplotlib.pyplot used to create figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc

seaborn is Python data visualization library based on matplotlib.

**Setup**:

Download the TAR/ZIP file from google to a location where you want to save.

Double-click the icon labelling the file python-3.7.9-amd64.exe.

A Python 3.7.9 (64-bit) Setup pop-up window will appear.

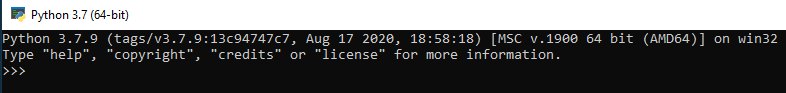
1. Ensure that the Install launcher for all users (recommended) and the Add Python 3.9 to PATH checkboxes at the bottom are checked.
2. Highlight the Install Now (or Upgrade Now) message, and then click it.
3. Click the Yes button.

During installation, it will show the various components it is installing and move the progress bar towards completion. Soon, a new Python 3.7.9 (64-bit) Setup pop-up window will appear with a Setup was successfully message.

1. Click the Close button.

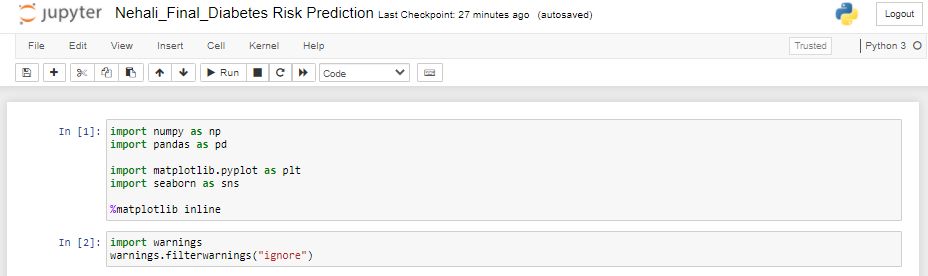
Python should now be installed.

To confirm installation, open the .exe file from the location

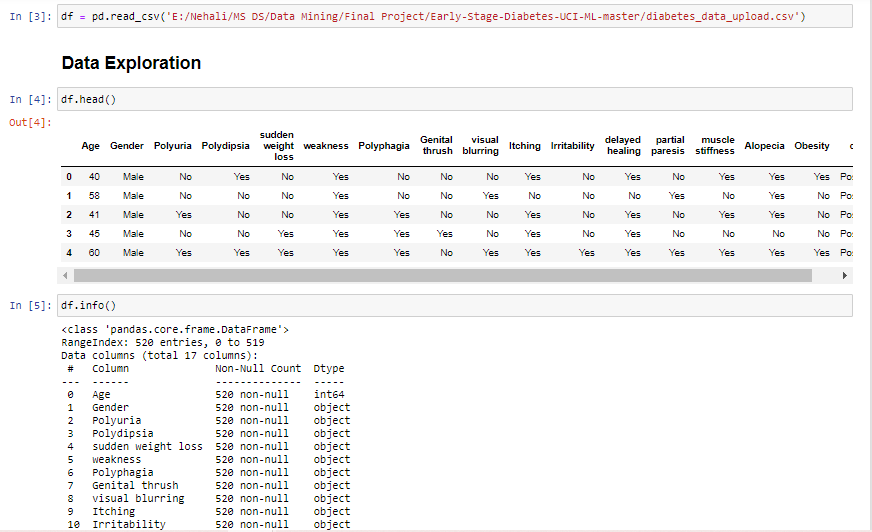


**Installation:**

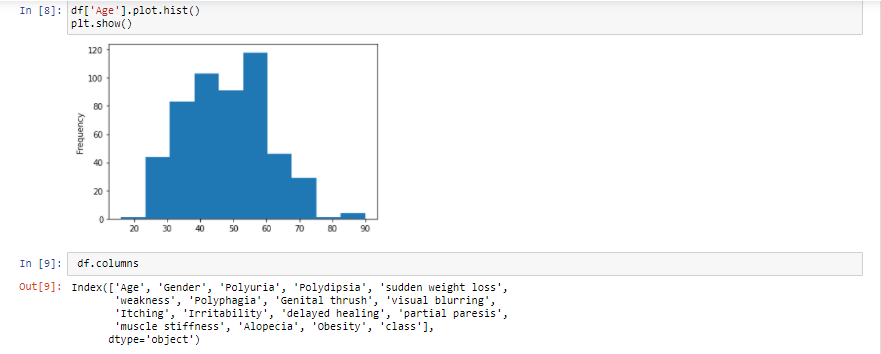
1. Install the necessary packages numpy, pandas, csv, matplotlib.pyplot, seaborn
2. Once the packages are installed start with the imports



1. Once the libraries are imported, import the dataset and explore the dataset

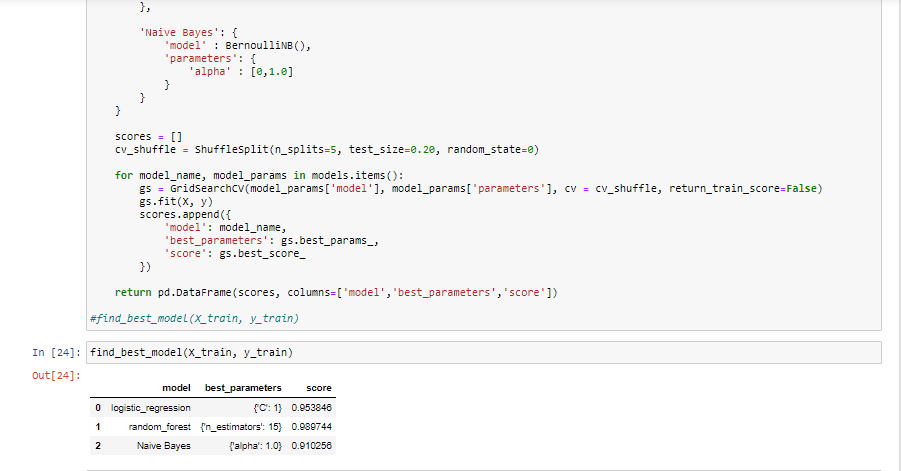


1. Generate a histogram to classify the data according to the age to determine a pattern.



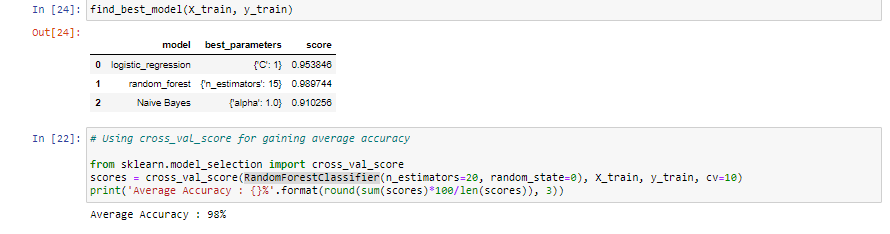
1. Start building the models to using the sklearn package.



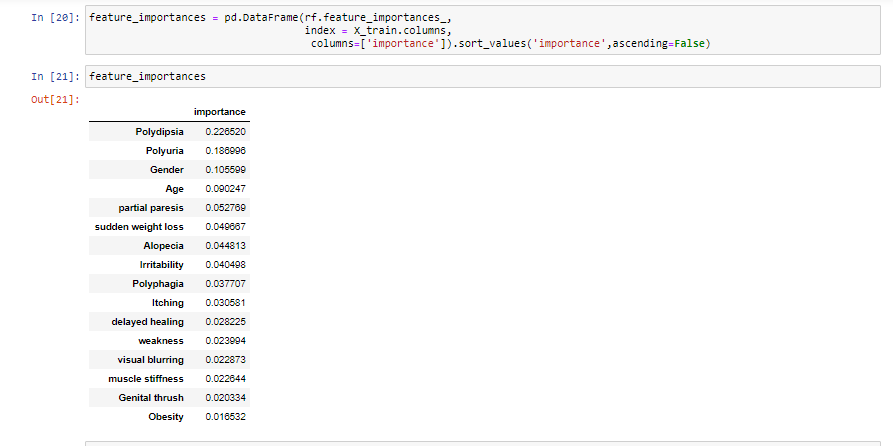


**Project Output:**

The following are the datasets and the generated output from the program.







**Program Files**

*Nehali\_Final\_Diabetes Risk Prediction*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('E:/Nehali/MS DS/Data Mining/Final Project/Early-Stage-Diabetes-UCI-ML-master/diabetes\_data\_upload.csv')

df.head()

df.info()

df['Gender'] = df['Gender'].apply(str)

df['class'].value\_counts(), df['Gender'].value\_counts()

df['Age'].plot.hist()

plt.show()

df.columns

from sklearn.model\_selection import train\_test\_split

X = df.drop(['class'], axis='columns')

y = df['class']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=50)

#Using GridSearchCV to find the best algorithm for this problem

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import ShuffleSplit

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import BernoulliNB

# Creating a function to calculate best model for this problem

def find\_best\_model(X, y):

models = {

'logistic\_regression': {

'model': LogisticRegression(solver='lbfgs', multi\_class='auto'),

'parameters': {

'C': [1,5,10]

}

},

'random\_forest': {

'model': RandomForestClassifier(criterion='gini'),

'parameters': {

'n\_estimators': [10,15,20,50,100,200]

}

},

'Naive Bayes': {

'model' : BernoulliNB(),

'parameters': {

'alpha' : [0,1.0]

}

}

}

scores = []

cv\_shuffle = ShuffleSplit(n\_splits=5, test\_size=0.20, random\_state=0)

for model\_name, model\_params in models.items():

gs = GridSearchCV(model\_params['model'], model\_params['parameters'], cv = cv\_shuffle, return\_train\_score=False)

gs.fit(X, y)

scores.append({

'model': model\_name,

'best\_parameters': gs.best\_params\_,

'score': gs.best\_score\_

})

return pd.DataFrame(scores, columns=['model','best\_parameters','score'])

#find\_best\_model(X\_train, y\_train)

find\_best\_model(X\_train, y\_train)

# Using cross\_val\_score for gaining average accuracy

from sklearn.model\_selection import cross\_val\_score

scores = cross\_val\_score(RandomForestClassifier(n\_estimators=20, random\_state=0), X\_train, y\_train, cv=10)

print('Average Accuracy : {}%'.format(round(sum(scores)\*100/len(scores)), 3))

# Creating Random Forest Model

rf = RandomForestClassifier(n\_estimators=100, random\_state=69)

rf.fit(X\_train, y\_train)

# Creating a confusion matrix

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

y\_pred = rf.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, cbar=False, annot=True)

plt.show()

# Classification Report

print(classification\_report(y\_test, y\_pred))

feature\_importances = pd.DataFrame(rf.feature\_importances\_,

index = X\_train.columns,

columns=['importance']).sort\_values('importance',ascending=False)

feature\_importances