Artificial Intelligence (Thursday)

Project: Artificial Intelligence techniques with conventional and Blockchain merger analysis.

Written By: Muhammad Raheel

Student ID# 62793

Submitted To: Sir Siraj Munir

All codes and datasets which are used in this project can be found at:

https://github.com/raheel4033/artifical-intelligence

Table of Contents:

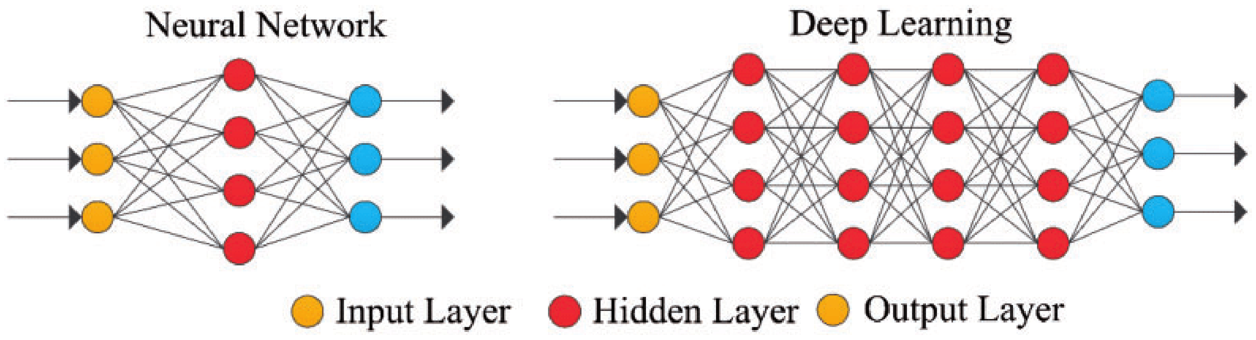
Introduction:

Using block chain as a database is one of the recent talks regarding the authenticity of the databases as I have gone through the conventional way of data handling using the normal databases where anomalies are dealt with the day to day measure and we have to go through certain options to make it work like taking mean or average between the values to make it work.

BIGCHAINDB:

BigChainDb is one of the recent databases which is released with a version 2.0 at the very least, it is written on the mongoDB with the flavor of decentralized system as blockchain itself is.Once the data is pushed in the bigchaindb then it cannot be altered and it is immutable. All the data in the bigchaindb is considered to be asset. BigChainDb supports both the traditional stack which is having the simple working mechanism and also the blockchain stack which is with the decentralized form of information.As it is using the mongoDb at the back end it has both the blockchain features and the distributed stack.

Most of the efficiency of the dataset which is being used and applied the algorithms on is dependent on the data rather than the algorithms. Algorithms are just commodities and in that manner the concept of deep learning comes to existence where the layers are provided one over the other rather than the neural network just make the efficiency better and better.



Having data is one of the largest resources you can have now a days as most of the organizations only do have the data and most the AI startups have the AI expertise but not much data to play with, only the giants like Facebook and google having access to the data can work through the different paradigms. But, in all this scenario why the blockchain and is important?

There must be a check and balance to everything right from what is being deployed by the person to certain platform must have the following features which is not taken care mostly which includes:

1) Human right to data privacy and consent: Your data must not be misused, abused and worked upon on any platform.

2) Unlock Data: The form of data which is available must be open to all.

3) Identity and reputation over anonymity.

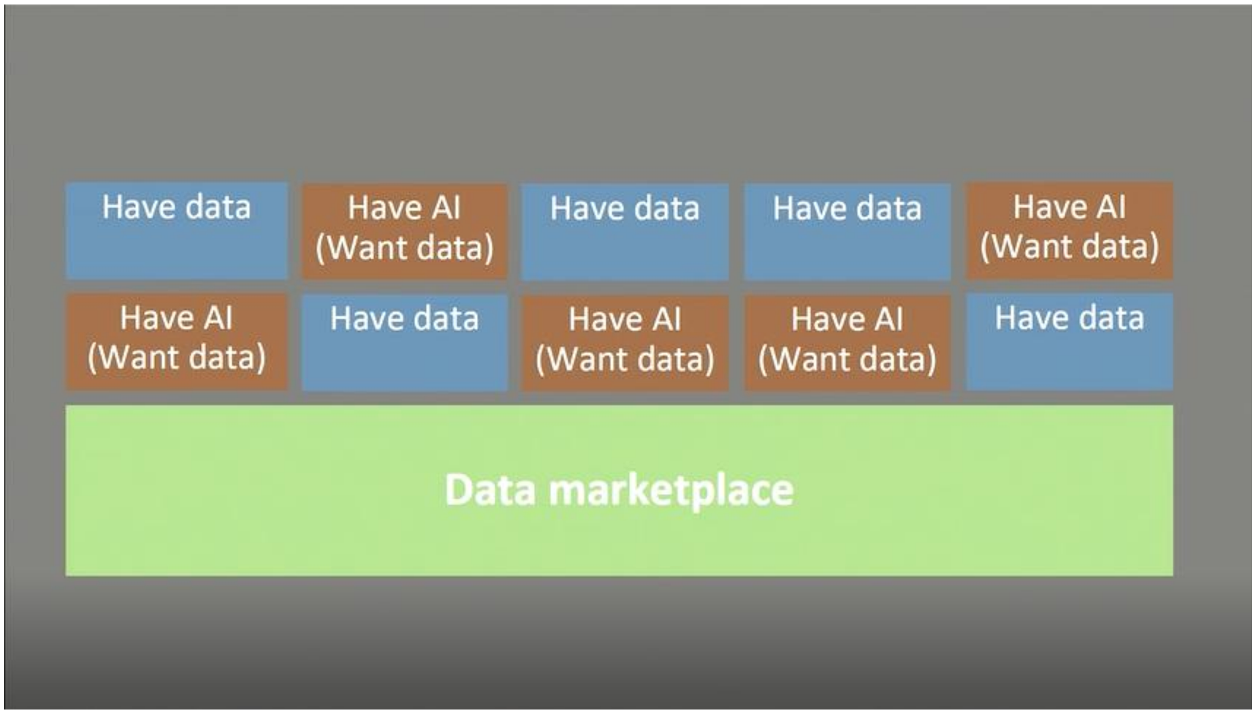
4) Spread of value power.

5) Sustainability.

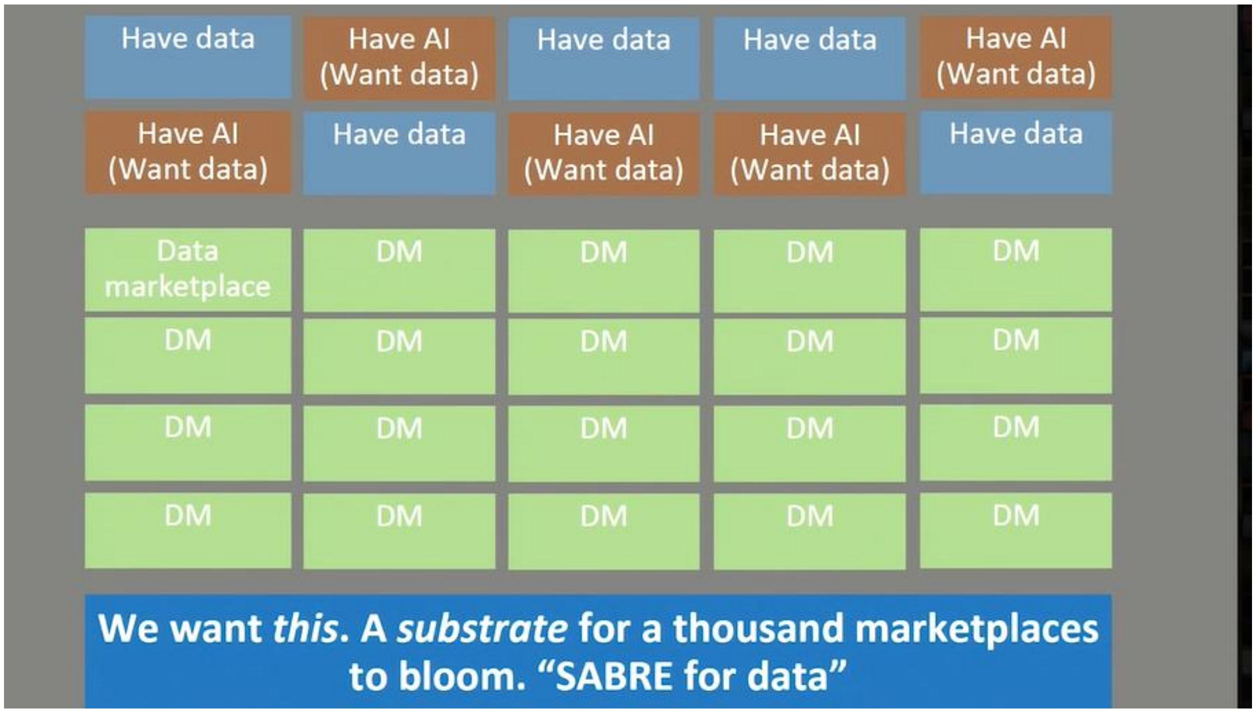
6) Resilience

Now what can be the solution?

One of the solutions could be to bring the enterprises having data and AI startups having expertise can be brought together just to make things work for both of them via a data marketplace.



But still by this technique the data won't be able to flow in such a way that the access is easier and efficient rather it will turned out to be in the same conventional way if not taken worked in the fashion that different places have different form of data in different distributions which make the access of user easier and efficient.



Most of the giants in financial, supply chain, health care and entertainment are spending in the research for their own decentralized platform for example Disney also had their own platform from the name of DragonChain. Problem starts with the scalability of the data since the data will be in large number.

So, Why Artificial Intelligence is important for blockchain?

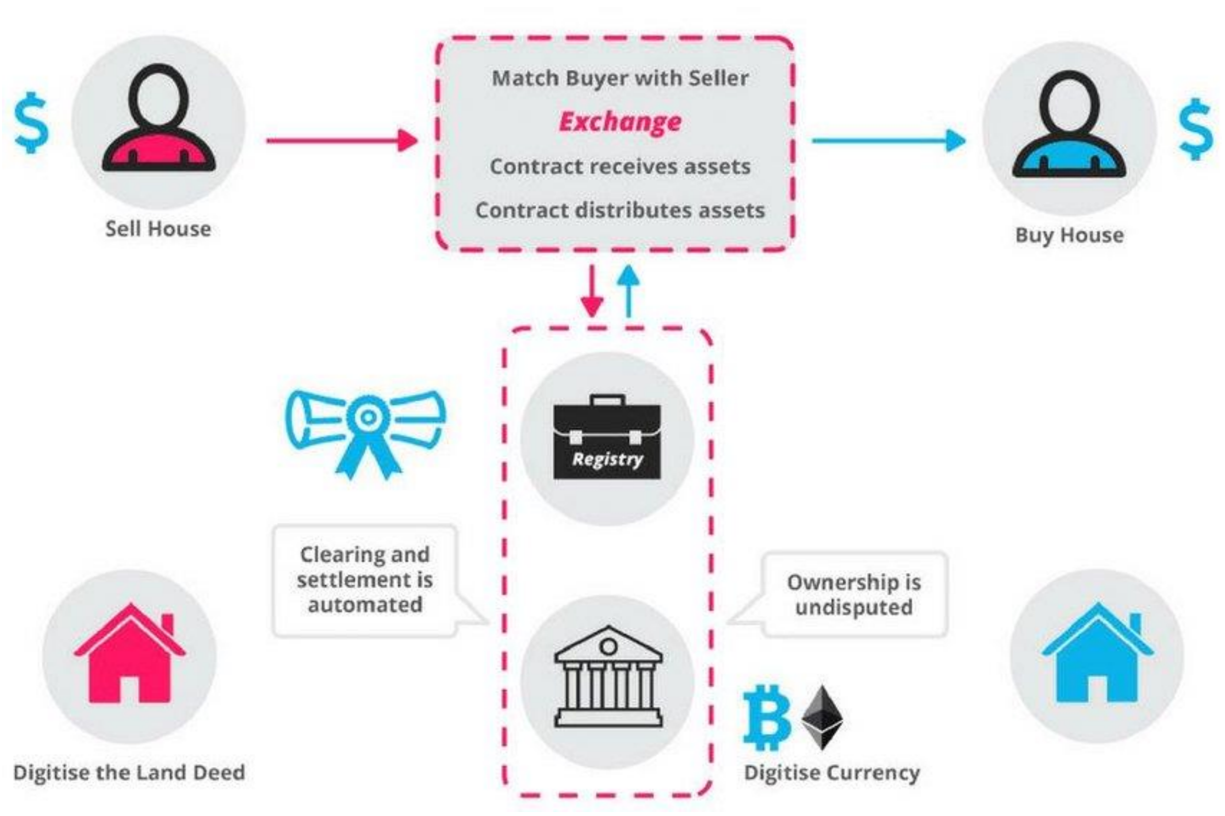
Artificial Intelligence based applications will help in blockchain to have the scenario where the data is complete, secure and trustable.

How to make it possible lets discuss.

What will be the applications of Blockchain in AI?

AI in smart Contracts.

Smart contracts are self-executing contracts for exchanging values. They varies from financial contracts, insurance contracts to all other kind of transactions.



Using AI in smart Contracts:

Self-sustained Artificial Intelligence systems can be used like it will allow the access to the set of initial rules like how to extract data gain insights form that data paying it for the services it will use. Then the system will be dependent on its own and can live with the network eternity

For this I have used the simple linear regression as a test to have an insight on the data it is discussed below:

Here is mentioned the importance of the data along with the data pre-processing steps required before applying the algorithms:

Importance of Data:

We are living in the world which is now consumed by the word called “DATA” as right from what we share on facebook to the text messages we sent the human is constructing the data in regular fashion to begin with:

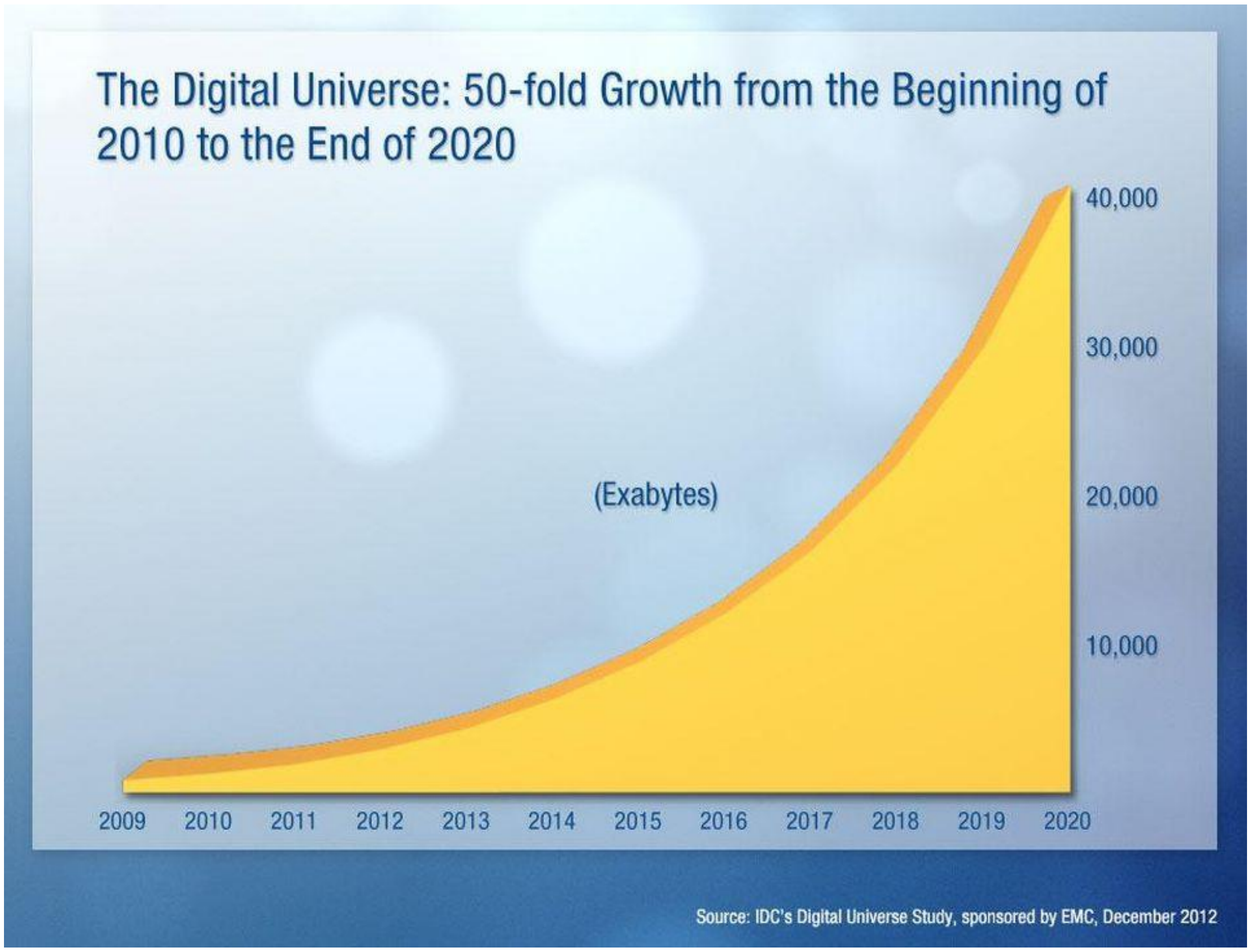
Since the dawn of time until 2005 the consumption of the data was 130 Exabyte.

Which gained up to 1200 Exabyte in the year 2010.

And around 2015 it had reached to 8000 Exabyte.

Now, as we live in 2020 it is estimated up to 40000 Exabyte.

So the growth of data itself in the decade and a half gives us what is going to meant in the future.



Step 1:

Install Anaconda

I have installed Anaconda (which is the package for scientific computing distribution) onto my machine as this aims to provide the platform to perform analysis.

Step 2:

Working on Data Preprocessing because if the data is not processed completely the model will not work properly.

Step 3:

Getting the dataset and analyze the variable which are dependent and independent on each other. Importing the libraries, a library is used to make the job easier and to have the access to the built-in functions. It is similar to the classes and the functions we gathers from those classes.

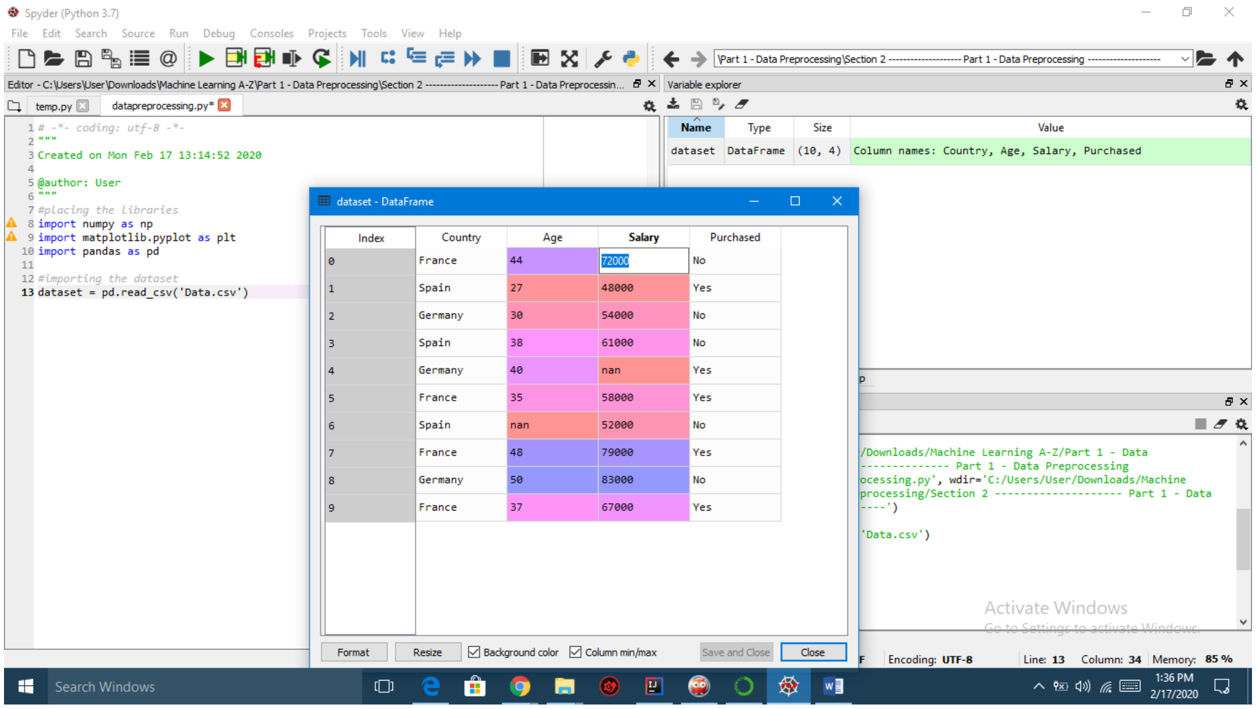
Three libraries for different computations has been included for data preprocessing.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

Step 4: Now I have changed the working directory and have read the dataset from with pd.read_cs v('Data.csv') as Data.csv is the file to which data is about to be explored.



```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 17 13:14:52 2020

@author: User
"""
#placing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the dataset
dataset = pd.read_csv('Data.csv')
```

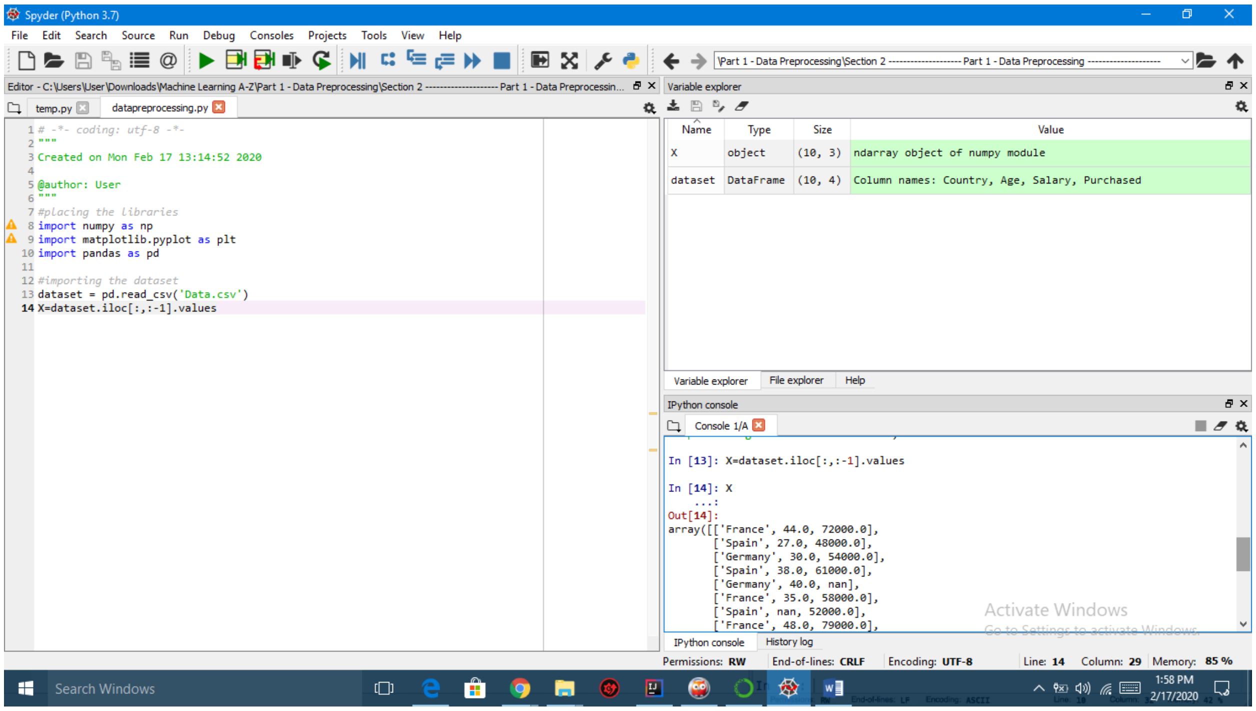
Name	Type	Size	Value
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

Index	Country	Age	Salary	Purchased
0	France	44	72000	No
1	Spain	27	48000	Yes
2	Germany	30	54000	No
3	Spain	38	61000	No
4	Germany	40	nan	Yes
5	France	35	58000	Yes
6	Spain	nan	52000	No
7	France	48	79000	Yes
8	Germany	50	83000	No
9	France	37	67000	Yes

```
/Downloads/Machine Learning A-Z/Part 1 - Data
--------------- Part 1 - Data Preprocessing
ocessing.py', wdir='C:/Users/User/Downloads/Machine
processing/Section 2 -------------------- Part 1 - Data
----')

'Data.csv')
```

In the above dataset (Country, age, salary, Purchased) are the variables in which the Country, age and salary are independent and Purchased is a dependent variable.

Now a new variable X is introduced in which the dataset will have the values along with the index in which we have used all the dependent variables and [:,:-1] (here -1 means that all the columns will be selected except the last one than we will gather all the information into an array.

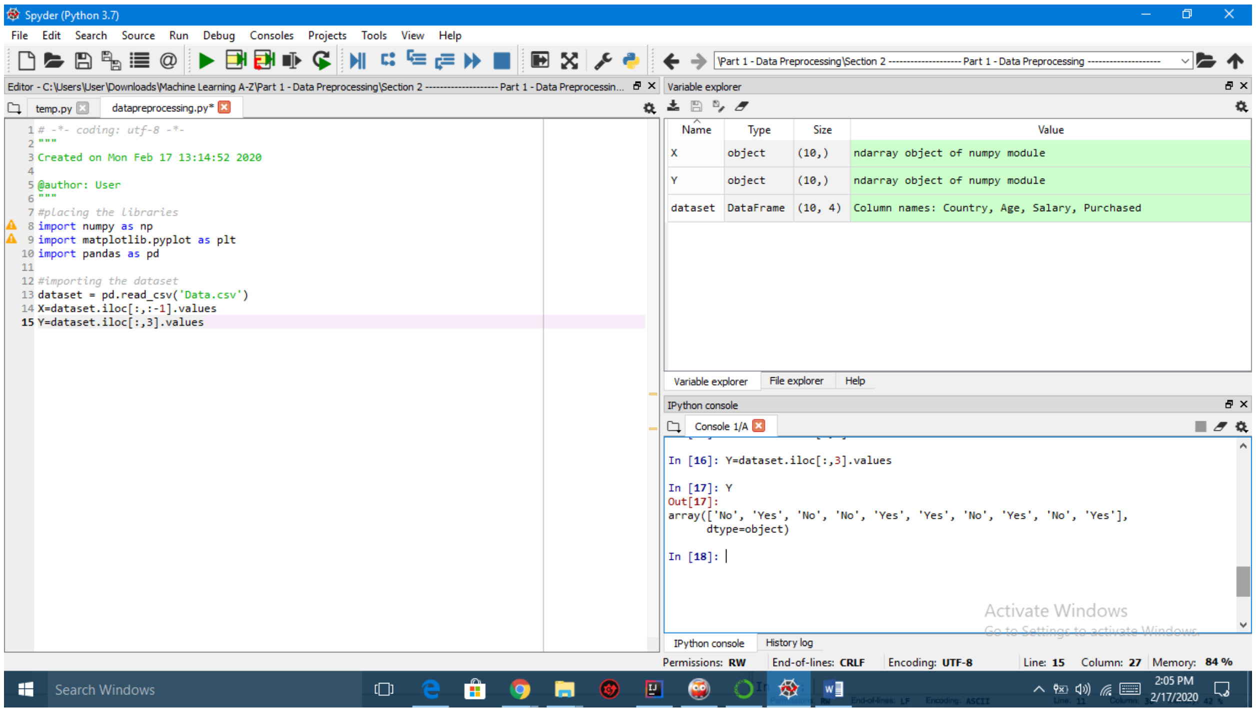


```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 17 13:14:52 2020

@author: User
"""
#placing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the dataset
dataset = pd.read_csv('Data.csv')
X=dataset.iloc[:,:-1].values
```

Name	Type	Size	Value
X	object	(10, 3)	ndarray object of numpy module
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

```
In [13]: X=dataset.iloc[:,:-1].values

In [14]: X
    ...:
Out[14]:
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, nan],
       ['France', 35.0, 58000.0],
       ['Spain', nan, 52000.0],
       ['France', 48.0, 79000.0],
```

Now another variable with the Y has been introduced with having only the details of thepurchase which are either ‘Yes’ or ‘No’.



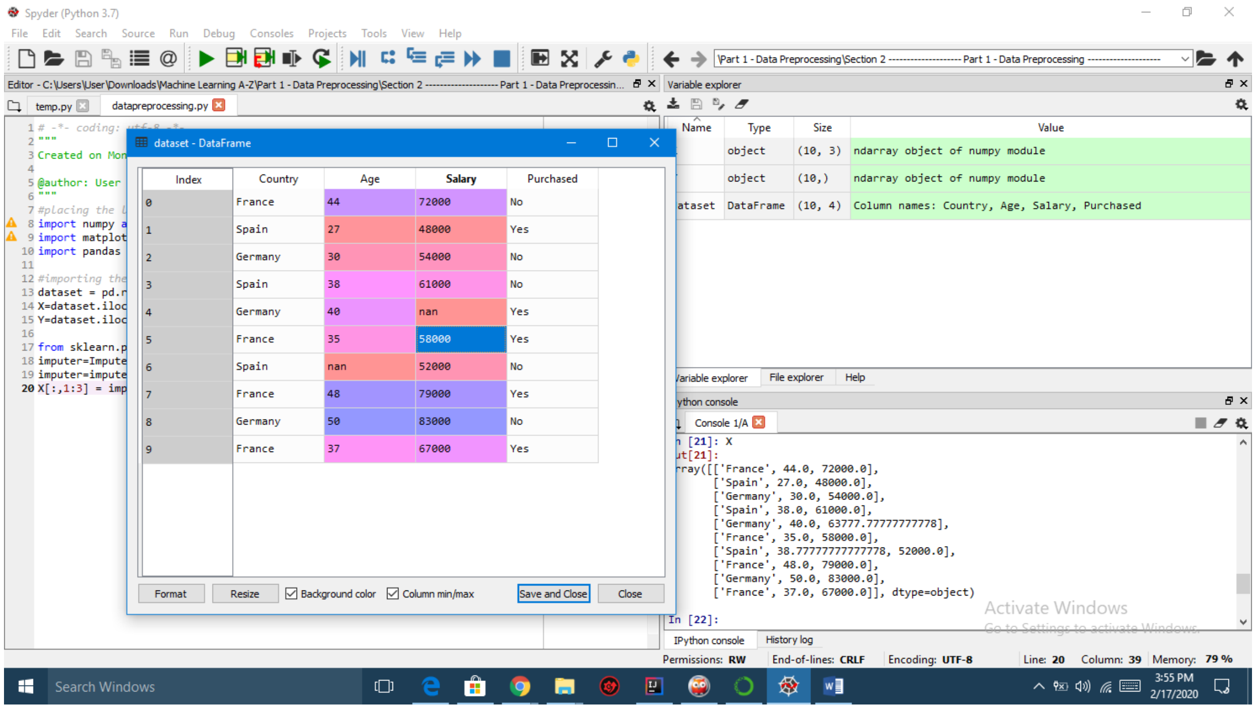
```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 17 13:14:52 2020

@author: User
"""
#placing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the dataset
dataset = pd.read_csv('Data.csv')
X=dataset.iloc[:,:-1].values
Y=dataset.iloc[:,3].values
```

Name	Type	Size	Value
X	object	(10,)	ndarray object of numpy module
Y	object	(10,)	ndarray object of numpy module
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

```
In [16]: Y=dataset.iloc[:,3].values

In [17]: Y
Out[17]:
array(['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'],
      dtype=object)

In [18]:
```

Step 5:

Missing Data will be comprised with the average mean of the column where the data is missing we will use missing_values as parameter and strategy keyword will be mean. In this way using this technique we have replaced Germany which had salary null will now be replaced by some will which is an average of all the values and similar is with the age column where the age is also the average of some value.



```
# -*- coding: utf-8 -*-
"""
Created on Mon

@author: User
"""
#placing the l
import numpy a
import matplot
import pandas

#importing the
dataset = pd.r
X=dataset.iloc
Y=dataset.iloc

from sklearn.p
imputer=Impute
imputer=impute
X[:,1:3] = imp
```

dataset - DataFrame

Index	Country	Age	Salary	Purchased
0	France	44	72000	No
1	Spain	27	48000	Yes
2	Germany	30	54000	No
3	Spain	38	61000	No
4	Germany	40	nan	Yes
5	France	35	58000	Yes
6	Spain	nan	52000	No
7	France	48	79000	Yes
8	Germany	50	83000	No
9	France	37	67000	Yes

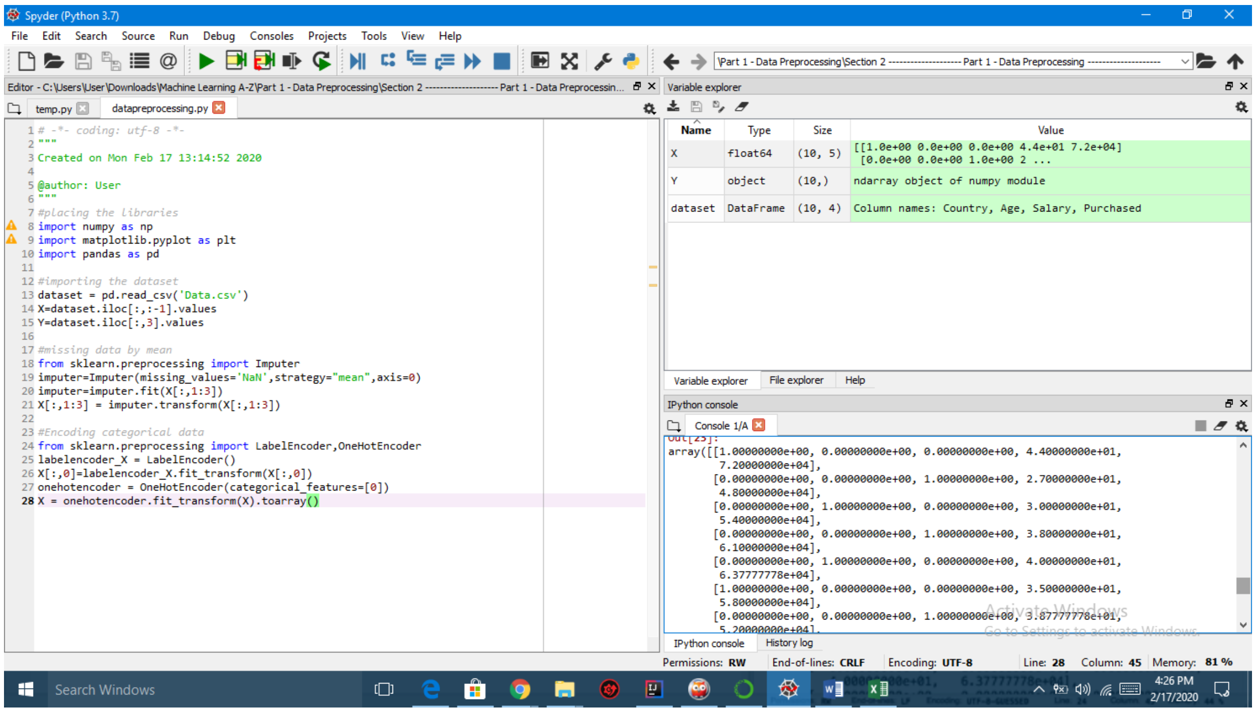
Name	Type	Size	Value
	object	(10, 3)	ndarray object of numpy module
	object	(10,)	ndarray object of numpy module
ataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

```
n [21]: X
ut[21]:
rray([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)

In [22]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 20 Column: 39 Memory: 79 %

Step 6:

Categorical Variable is usually fixed and has number of possible values assigned each individual to a unit of observation.


Editor - C:\Users\User\Downloads\Machine Learning A-Z\Part 1 - Data Preprocessing\Section 2 -------------------- Part 1 - Data Preprocessin...

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 17 13:14:52 2020

@author: User
"""
#placing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the dataset
dataset = pd.read_csv('Data.csv')
X=dataset.iloc[:,:-1].values
Y=dataset.iloc[:,3].values

#missing data by mean
from sklearn.preprocessing import Imputer
imputer=Imputer(missing_values='NaN',strategy="mean",axis=0)
imputer=imputer.fit(X[:,1:3])
X[:,1:3] = imputer.transform(X[:,1:3])

#Encoding categorical data
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
labelencoder_X = LabelEncoder()
X[:,0]=labelencoder_X.fit_transform(X[:,0])
onehotencoder = OneHotEncoder(categorical_features=[0])
X = onehotencoder.fit_transform(X).toarray()
```

Variable explorer

Name	Type	Size	Value
X	float64	(10, 5)	[[1.0e+00 0.0e+00 0.0e+00 4.4e+01 7.2e+04] [0.0e+00 0.0e+00 1.0e+00 2 ...
Y	object	(10,)	ndarray object of numpy module
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

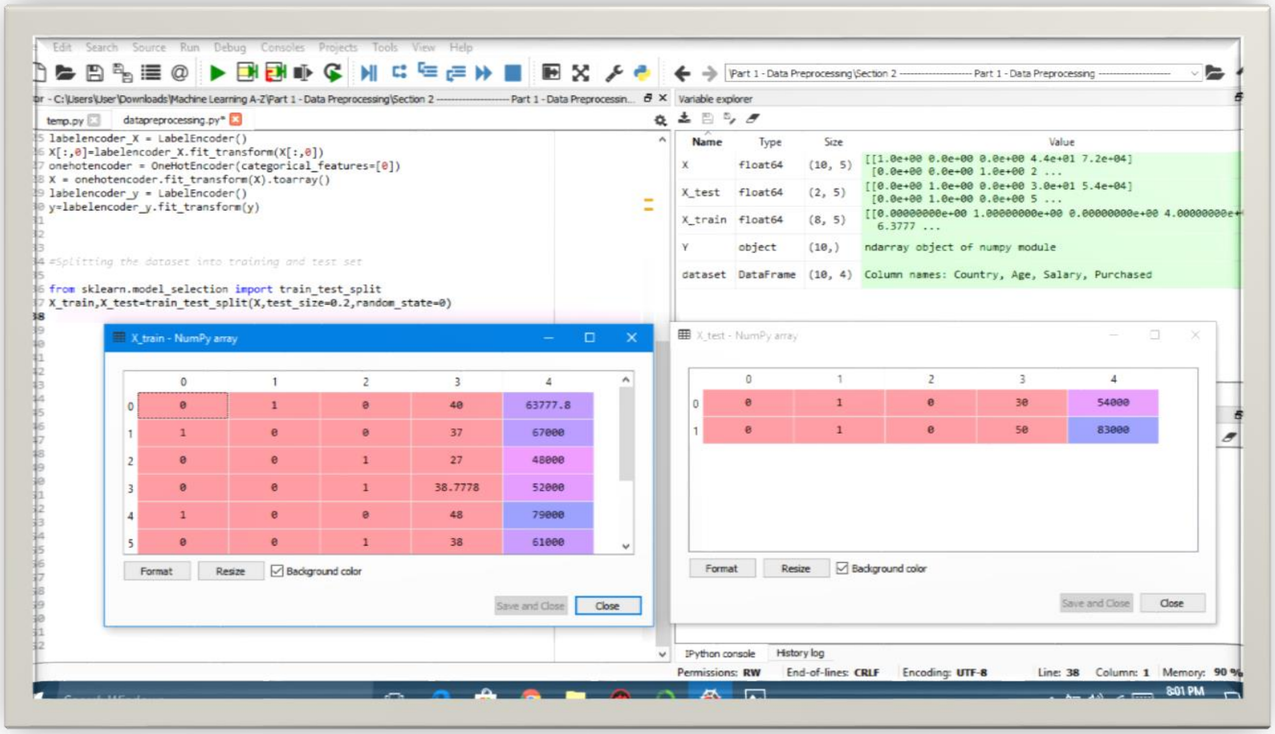
IPython console

```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.40000000e+01,
        7.20000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 2.70000000e+01,
        4.80000000e+04],
       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 3.00000000e+01,
        5.40000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.80000000e+01,
        6.10000000e+04],
       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 4.00000000e+01,
        6.37777778e+04],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.50000000e+01,
        5.80000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.87777778e+01,
        5.20000000e+04],
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 28 Column: 45 Memory: 81 %

Step 7:

Dataset has been spilt into X category so it has a train set and test set with test_size has been remained 0.2 means out of 10 values test_size will get its 2 values and remaining values will be in the training test.



```
labelencoder_X = LabelEncoder()
X[:,0]=labelencoder_X.fit_transform(X[:,0])
onehotencoder = OneHotEncoder(categorical_features=[0])
X = onehotencoder.fit_transform(X).toarray()
labelencoder_y = LabelEncoder()
y=labelencoder_y.fit_transform(y)



#Splitting the dataset into training and test set

from sklearn.model_selection import train_test_split
X_train,X_test=train_test_split(X,test_size=0.2,random_state=0)
```

Variable explorer

Name	Type	Size	Value
X	float64	(10, 5)	[[1.0e+00 0.0e+00 0.0e+00 4.4e+01 7.2e+04] [0.0e+00 0.0e+00 1.0e+00 2 ...
X_test	float64	(2, 5)	[[0.0e+00 1.0e+00 0.0e+00 3.0e+01 5.4e+04] [0.0e+00 1.0e+00 0.0e+00 5 ...
X_train	float64	(8, 5)	[[0.00000000e+00 1.00000000e+00 0.00000000e+00 4.00000000e+ 6.3777 ...
Y	object	(10,)	ndarray object of numpy module
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

X_train - NumPy array

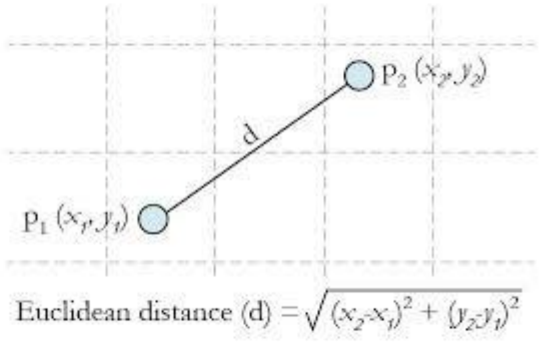
	0	1	2	3	4
0	0	1	0	40	63777.8
1	1	0	0	37	67000
2	0	0	1	27	48000
3	0	0	1	38.7778	52000
4	1	0	0	48	79000
5	0	0	1	38	61000

X_test - NumPy array

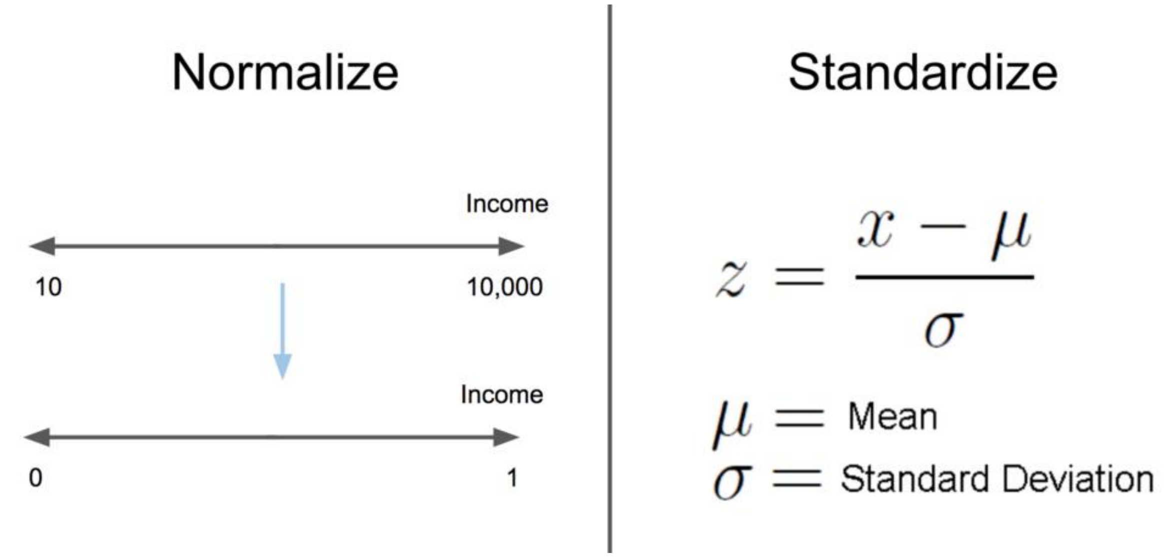
	0	1	2	3	4
0	0	1	0	30	54000
1	0	1	0	50	83000

Step 8:

Most of the machine learning models are based on the Euclidean distance which means there is a variation between our age variable range from 20’s to 50’s and similar goes with the salary as well.



So, there are two types of feature scaling first one is standardization and the other is normalization

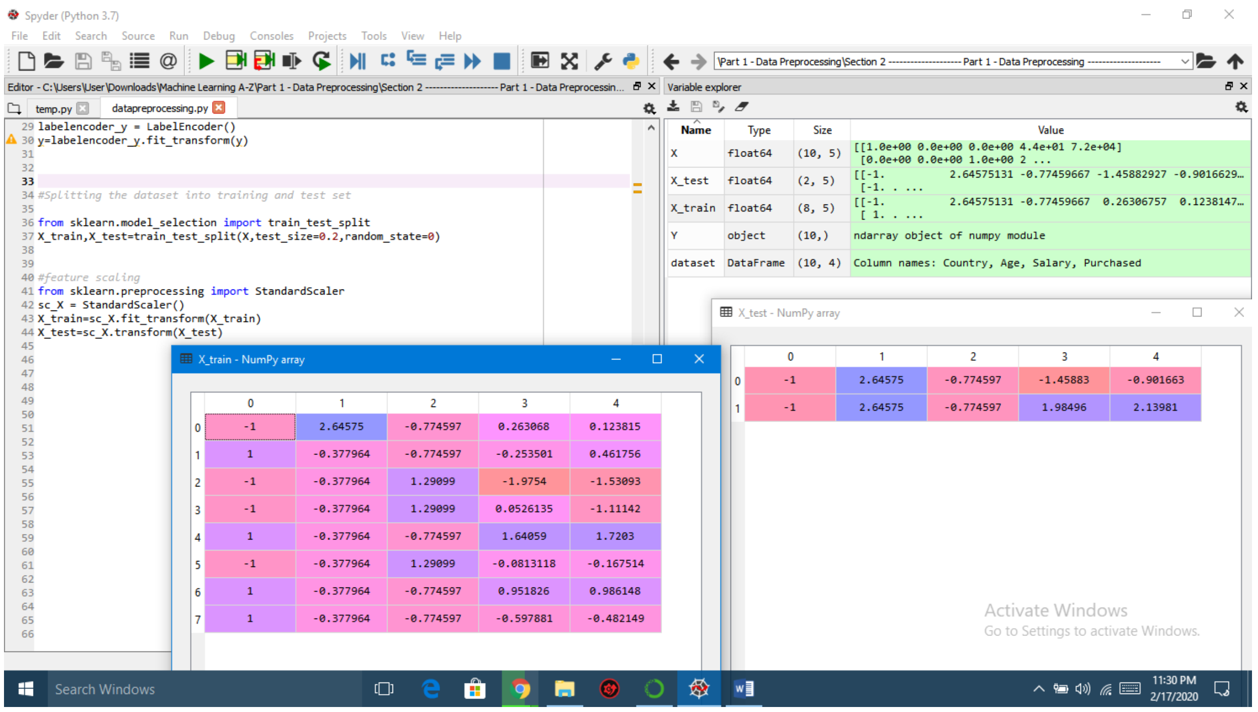


$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean

$\sigma =$ Standard Deviation

Now in the example below the data is featured scale as it is now range in between +1 and -1 which will be easy in the further discussion as on small range applying the machine learning algorithms will be easy.


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Users\User\Downloads\Machine Learning A-Z\Part 1 - Data Preprocessing\Section 2 ------------------- Part 1 - Data Preprocessin...
temp.py
datapreprocessing.py
Variable explorer
\Part 1 - Data Preprocessing\Section 2 ------------------- Part 1 - Data Preprocessing -------------------
X_train - NumPy array
X_test - NumPy array
Activate Windows
Go to Settings to activate Windows.
Search Windows
11:30 PM
2/17/2020

```
labelencoder_y = LabelEncoder()
y=labelencoder_y.fit_transform(y)



#Splitting the dataset into training and test set

from sklearn.model_selection import train_test_split
X_train,X_test=train_test_split(X,test_size=0.2,random_state=0)


#feature scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train=sc_X.fit_transform(X_train)
X_test=sc_X.transform(X_test)
```

Name	Type	Size	Value
X	float64	(10, 5)	[[1.0e+00 0.0e+00 0.0e+00 4.4e+01 7.2e+04] [0.0e+00 0.0e+00 1.0e+00 2 ...
X_test	float64	(2, 5)	[[-1. 2.64575131 -0.77459667 -1.45882927 -0.9016629… [-1. . ...
X_train	float64	(8, 5)	[[-1. 2.64575131 -0.77459667 0.26306757 0.1238147… [1. . ...
Y	object	(10,)	ndarray object of numpy module
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

X_train - NumPy array

	0	1	2	3	4
0	-1	2.64575	-0.774597	0.263068	0.123815
1	1	-0.377964	-0.774597	-0.253501	0.461756
2	-1	-0.377964	1.29099	-1.9754	-1.53093
3	-1	-0.377964	1.29099	0.0526135	-1.11142
4	1	-0.377964	-0.774597	1.64059	1.7203
5	-1	-0.377964	1.29099	-0.0813118	-0.167514
6	1	-0.377964	-0.774597	0.951826	0.986148
7	1	-0.377964	-0.774597	-0.597881	-0.482149

X_test - NumPy array

	0	1	2	3	4
0	-1	2.64575	-0.774597	-1.45883	-0.901663
1	-1	2.64575	-0.774597	1.98496	2.13981

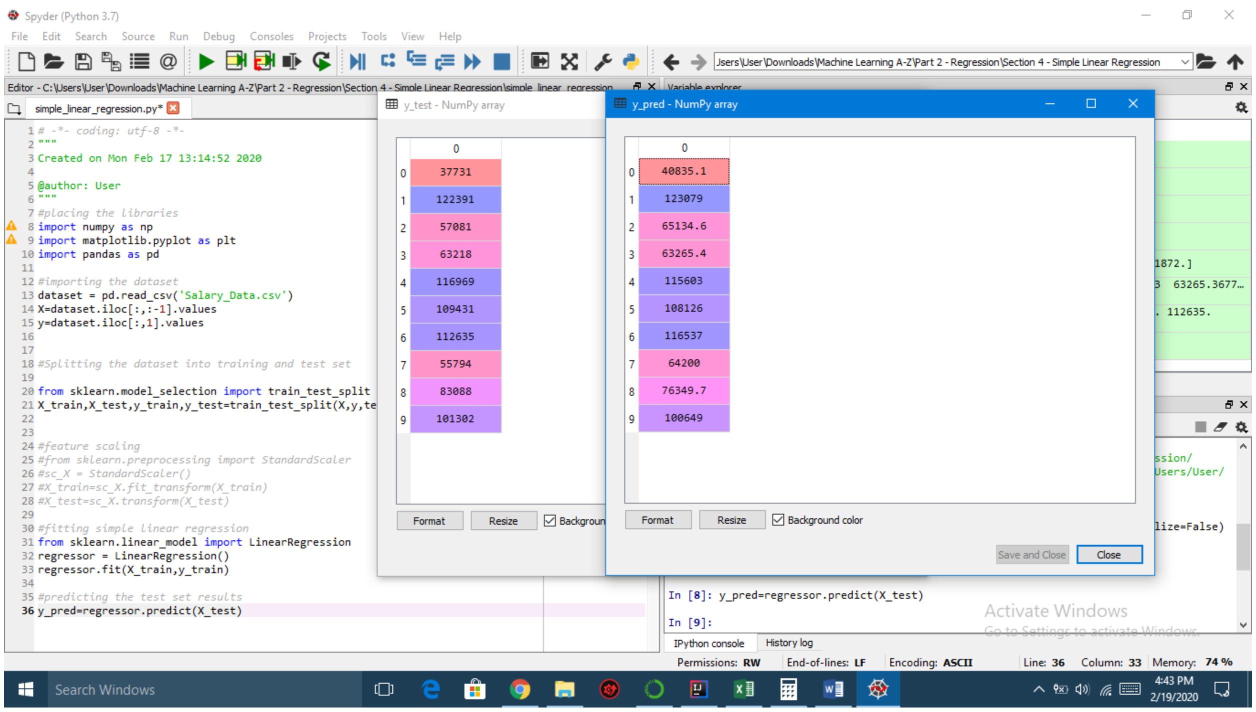
Simple Linear Regression:

It is a machine learning algorithms which works on the simple formula.

y=b0+b1x

Here y is dependent variable and x is independent variable with the + as coefficient and b0 is constant.

The predicted variable y_pred has predicted the outcome based on this algorithm



```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 17 13:14:52 2020

@author: User
"""
#placing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,1].values


#Splitting the dataset into training and test set

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,te


#feature scaling
#from sklearn.preprocessing import StandardScaler
#sc_X = StandardScaler()
#X_train=sc_X.fit_transform(X_train)
#X_test=sc_X.transform(X_test)

#fitting simple linear regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)

#predicting the test set results
y_pred=regressor.predict(X_test)
```

y_test - NumPy array

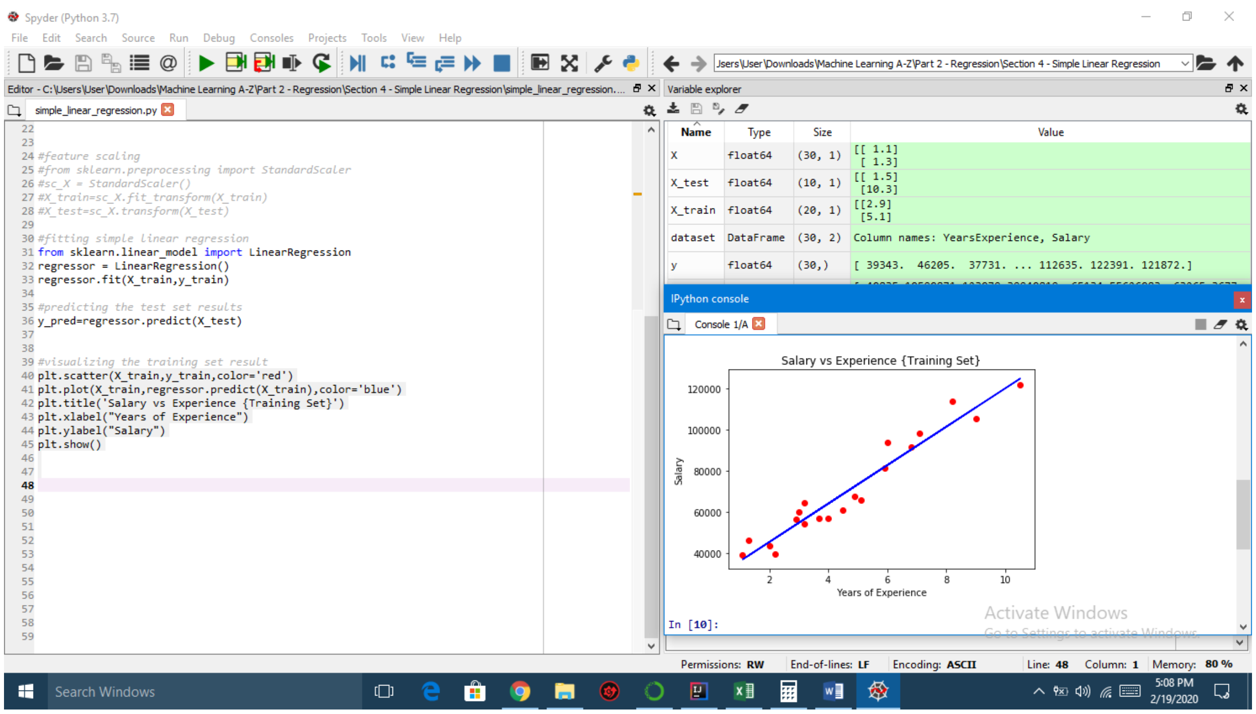
	0
0	37731
1	122391
2	57081
3	63218
4	116969
5	109431
6	112635
7	55794
8	83088
9	101302

y_pred - NumPy array

	0
0	40835.1
1	123079
2	65134.6
3	63265.4
4	115603
5	108126
6	116537
7	64200
8	76349.7
9	100649

```
In [8]: y_pred=regressor.predict(X_test)

In [9]:
```

Here the final visualization of the data using the plt with its different functions with x-axis being the years of experience and y-axis as the salary with original salary being the red point and the predicted salary is the what is being on the regression line. Here is the code and its output.

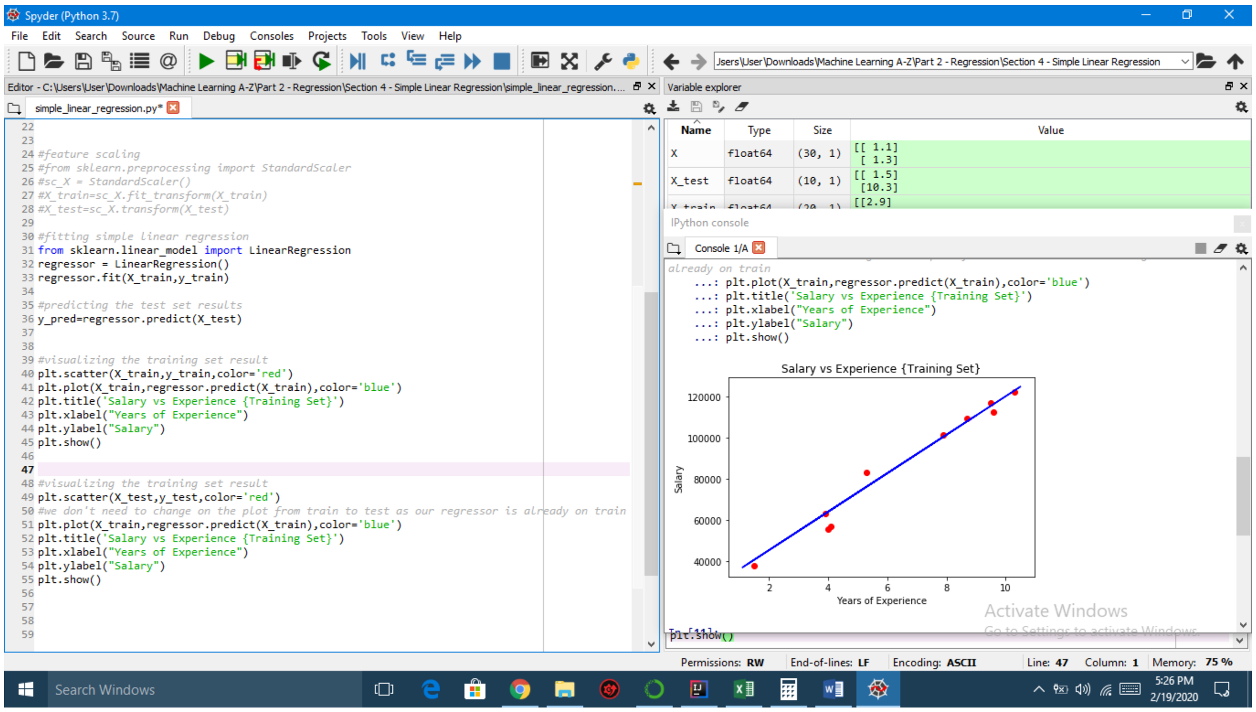


```
#feature scaling
#from sklearn.preprocessing import StandardScaler
#sc_X = StandardScaler()
#X_train=sc_X.fit_transform(X_train)
#X_test=sc_X.transform(X_test)

#fitting simple linear regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)

#predicting the test set results
y_pred=regressor.predict(X_test)


#visualizing the training set result
plt.scatter(X_train,y_train,color='red')
plt.plot(X_train,regressor.predict(X_train),color='blue')
plt.title('Salary vs Experience {Training Set}')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()
```

Name	Type	Size	Value
X	float64	(30, 1)	[[1.1] [1.3]
X_test	float64	(10, 1)	[[1.5] [10.3]
X_train	float64	(20, 1)	[[2.9] [5.1]
dataset	DataFrame	(30, 2)	Column names: YearsExperience, Salary
y	float64	(30,)	[39343. 46205. 37731. ... 112635. 122391. 121872.]

```
In [10]:
```

Now making the visualization on the test set so we have to make changes in the scatter function only where I place X_test and y_test instead of X_train and y_train



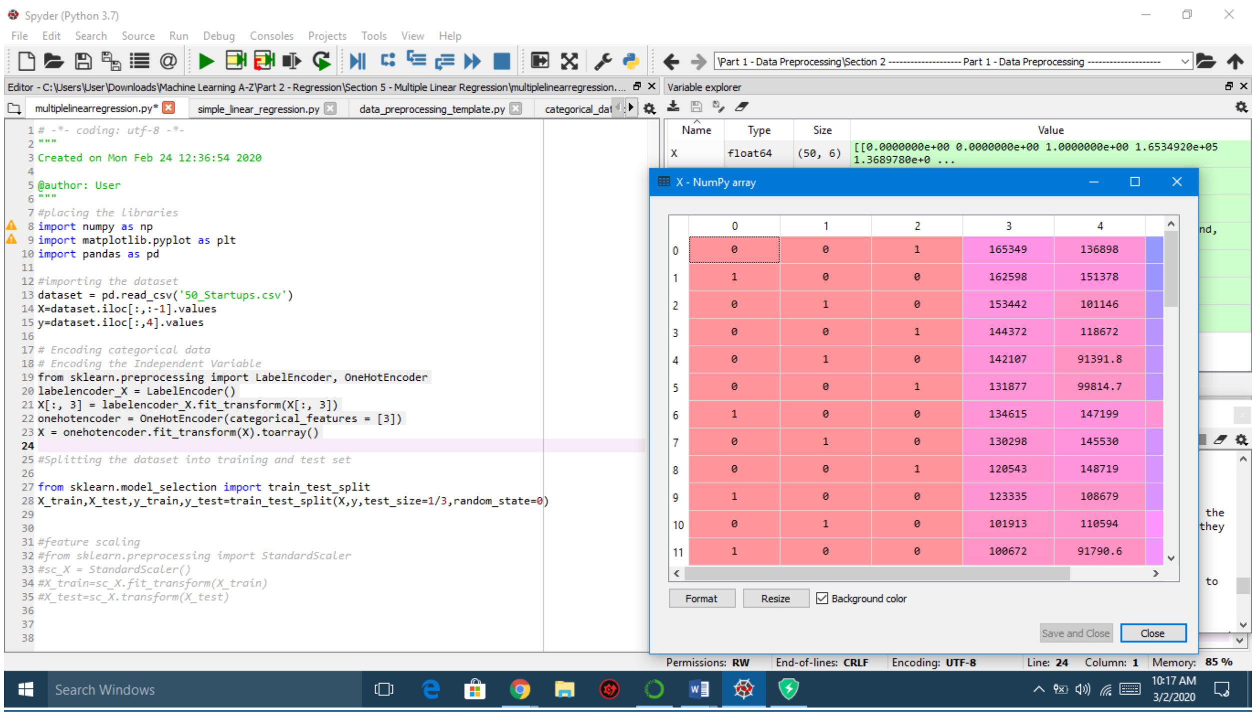
```
#feature scaling
#from sklearn.preprocessing import StandardScaler
#sc_X = StandardScaler()
#X_train=sc_X.fit_transform(X_train)
#X_test=sc_X.transform(X_test)

#fitting simple linear regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)

#predicting the test set results
y_pred=regressor.predict(X_test)


#visualizing the training set result
plt.scatter(X_train,y_train,color='red')
plt.plot(X_train,regressor.predict(X_train),color='blue')
plt.title('Salary vs Experience {Training Set}')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()


#visualizing the training set result
plt.scatter(X_test,y_test,color='red')
#we don't need to change on the plot from train to test as our regressor is already on train
plt.plot(X_train,regressor.predict(X_train),color='blue')
plt.title('Salary vs Experience {Training Set}')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()
```

```
already on train
   ...: plt.plot(X_train,regressor.predict(X_train),color='blue')
   ...: plt.title('Salary vs Experience {Training Set}')
   ...: plt.xlabel("Years of Experience")
   ...: plt.ylabel("Salary")
   ...: plt.show()
```

Similarly for the Multiple linear Regression Algorithms:

In Multiple Linear regression the dependent variables increases as compared to the simple linear regression:

y=b0+b1x+b2x+b3x

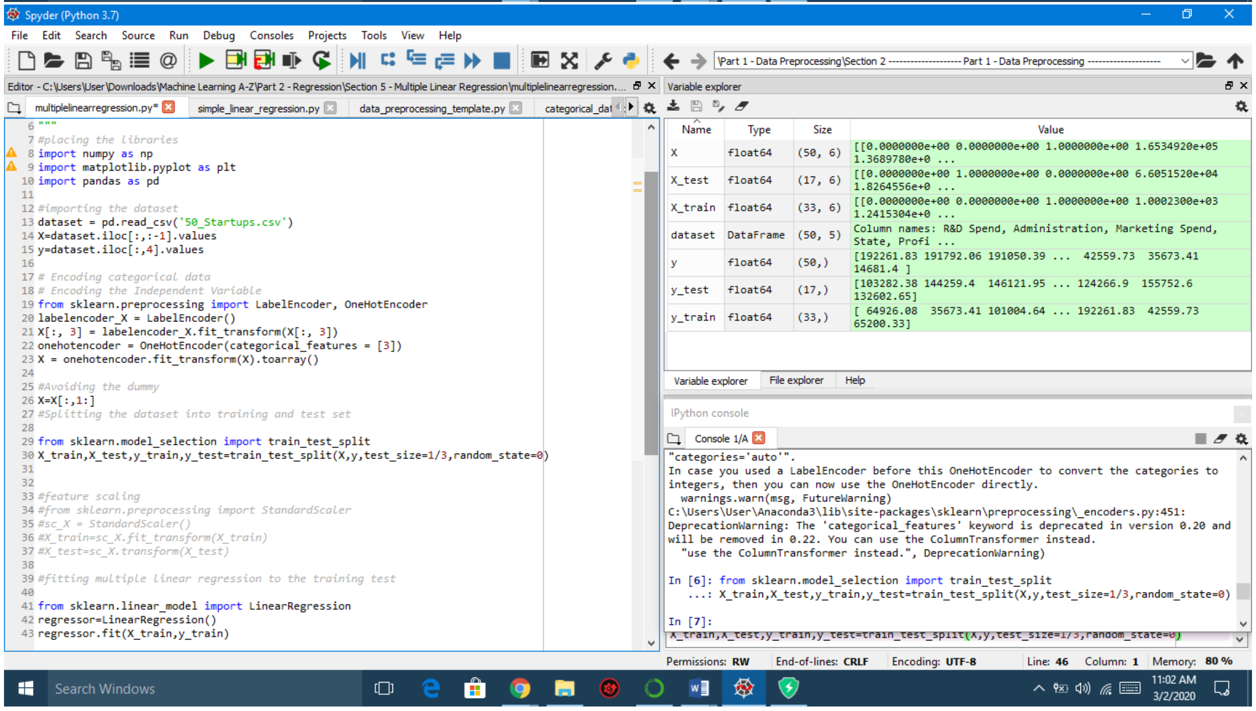


```python
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 24 12:36:54 2020

@author: User
"""
#placing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the dataset
dataset = pd.read_csv('50_Startups.csv')
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,4].values

# Encoding categorical data
# Encoding the Independent Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 3] = labelencoder_X.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()

#Splitting the dataset into training and test set

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/3,random_state=0)


#feature scaling
#from sklearn.preprocessing import StandardScaler
#sc_X = StandardScaler()
#X_train=sc_X.fit_transform(X_train)
#X_test=sc_X.transform(X_test)
```

Name	Type	Size	Value
X	float64	(50, 6)	[[0.0000000e+00 0.0000000e+00 1.0000000e+00 1.6534920e+05 1.3689780e+0 ...

X - NumPy array

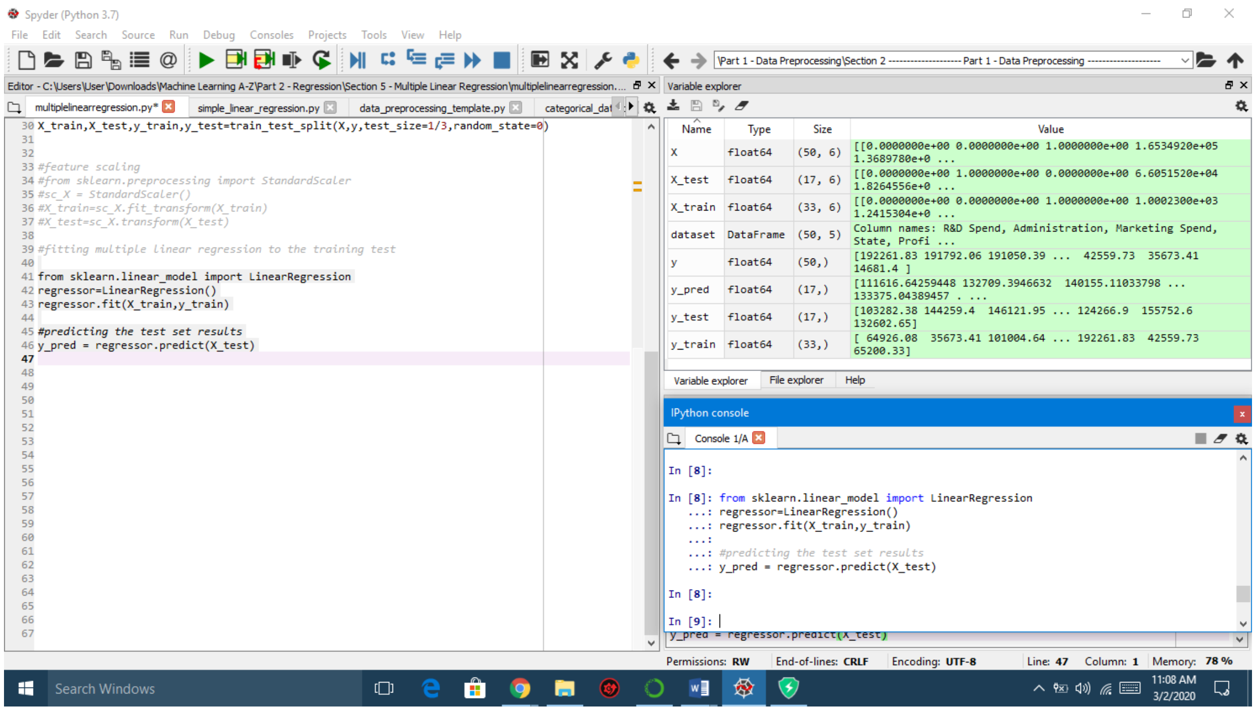
	0	1	2	3	4
0	0	0	1	165349	136898
1	1	0	0	162598	151378
2	0	1	0	153442	101146
3	0	0	1	144372	118672
4	0	1	0	142107	91391.8
5	0	0	1	131877	99814.7
6	1	0	0	134615	147199
7	0	1	0	130298	145530
8	0	0	1	120543	148719
9	1	0	0	123335	108679
10	0	1	0	101913	110594
11	1	0	0	100672	91790.6



```python
"""
#placing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the dataset
dataset = pd.read_csv('50_Startups.csv')
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,4].values

# Encoding categorical data
# Encoding the Independent Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 3] = labelencoder_X.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()

#Avoiding the dummy
X=X[:,1:]
#Splitting the dataset into training and test set

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/3,random_state=0)


#feature scaling
#from sklearn.preprocessing import StandardScaler
#sc_X = StandardScaler()
#X_train=sc_X.fit_transform(X_train)
#X_test=sc_X.transform(X_test)

#fitting multiple linear regression to the training test

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
```

Name	Type	Size	Value
X	float64	(50, 6)	[[0.0000000e+00 0.0000000e+00 1.0000000e+00 1.6534920e+05 1.3689780e+0 ...
X_test	float64	(17, 6)	[[0.0000000e+00 1.0000000e+00 0.0000000e+00 6.6051520e+04 1.8264556e+0 ...
X_train	float64	(33, 6)	[[0.0000000e+00 0.0000000e+00 1.0000000e+00 1.0002300e+03 1.2415304e+0 ...
dataset	DataFrame	(50, 5)	Column names: R&D Spend, Administration, Marketing Spend, State, Profi ...
y	float64	(50,)	[192261.83 191792.06 191050.39 ... 42559.73 35673.41 14681.4]
y_test	float64	(17,)	[103282.38 144259.4 146121.95 ... 124266.9 155752.6 132602.65]
y_train	float64	(33,)	[64926.08 35673.41 101004.64 ... 192261.83 42559.73 65200.33]

```
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to
integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\User\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:451:
DeprecationWarning: The 'categorical_features' keyword is deprecated in version 0.20 and
will be removed in 0.22. You can use the ColumnTransformer instead.
  "use the ColumnTransformer instead.", DeprecationWarning)

In [6]: from sklearn.model_selection import train_test_split
   ...: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/3,random_state=0)

In [7]:
```

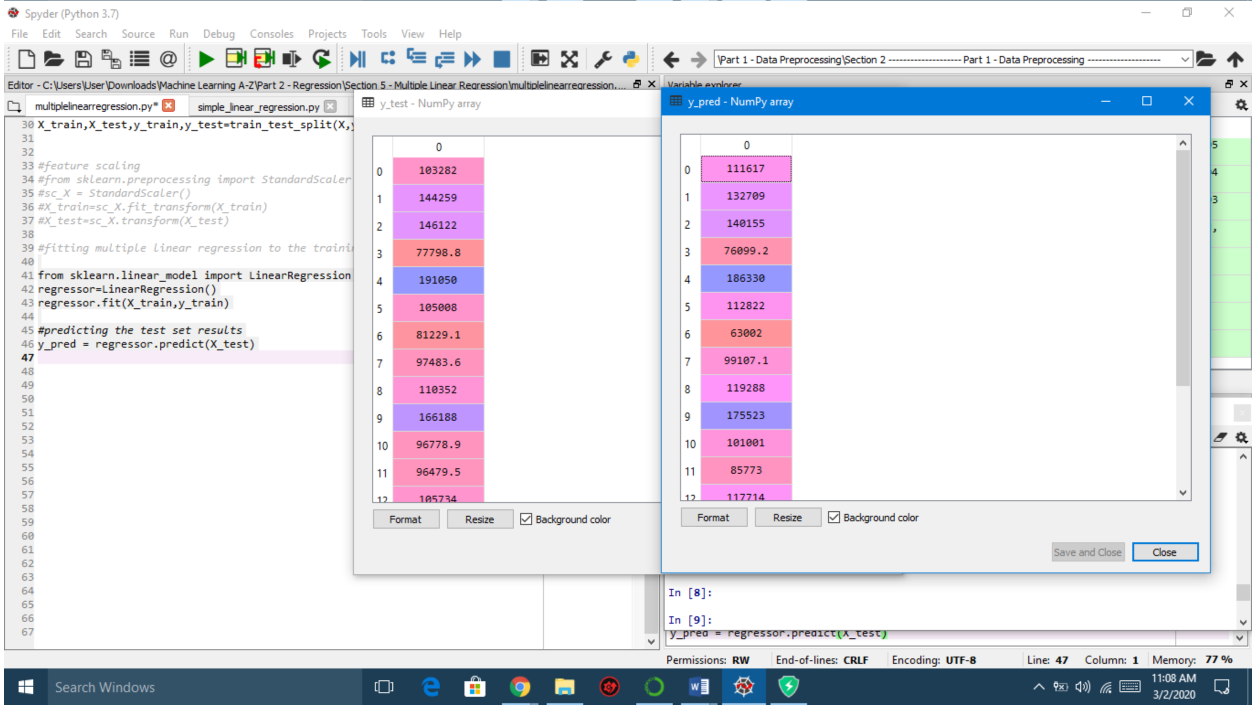


```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/3,random_state=0)


#feature scaling
#from sklearn.preprocessing import StandardScaler
#sc_X = StandardScaler()
#X_train=sc_X.fit_transform(X_train)
#X_test=sc_X.transform(X_test)

#fitting multiple linear regression to the training test

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)

#predicting the test set results
y_pred = regressor.predict(X_test)
```

Name	Type	Size	Value
X	float64	(50, 6)	[[0.0000000e+00 0.0000000e+00 1.0000000e+00 1.6534920e+05 1.3689780e+0 ...
X_test	float64	(17, 6)	[[0.0000000e+00 1.0000000e+00 0.0000000e+00 6.6051520e+04 1.8264556e+0 ...
X_train	float64	(33, 6)	[[0.0000000e+00 0.0000000e+00 1.0000000e+00 1.0002300e+03 1.2415304e+0 ...
dataset	DataFrame	(50, 5)	Column names: R&D Spend, Administration, Marketing Spend, State, Profi ...
y	float64	(50,)	[192261.83 191792.06 191050.39 ... 42559.73 35673.41 14681.4]
y_pred	float64	(17,)	[111616.64259448 132709.3946632 140155.11033798 ... 133375.04389457 . ...
y_test	float64	(17,)	[103282.38 144259.4 146121.95 ... 124266.9 155752.6 132602.65]
y_train	float64	(33,)	[64926.08 35673.41 101004.64 ... 192261.83 42559.73 65200.33]

```
In [8]:

In [8]: from sklearn.linear_model import LinearRegression
   ...: regressor=LinearRegression()
   ...: regressor.fit(X_train,y_train)
   ...:
   ...: #predicting the test set results
   ...: y_pred = regressor.predict(X_test)

In [8]:

In [9]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 47 Column: 1 Memory: 78 %



y_test - NumPy array

	0
0	103282
1	144259
2	146122
3	77798.8
4	191050
5	105008
6	81229.1
7	97483.6
8	110352
9	166188
10	96778.9
11	96479.5
12	105734

y_pred - NumPy array

	0
0	111617
1	132709
2	140155
3	76099.2
4	186330
5	112822
6	63002
7	99107.1
8	119288
9	175523
10	101001
11	85773
12	117714

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 47 Column: 1 Memory: 77 %

K-Means:

It's an example of clustering algorithms which means grouping.

Step 1: Chose number of cluster K.

Step 2: Select Centroid from scatter plot.

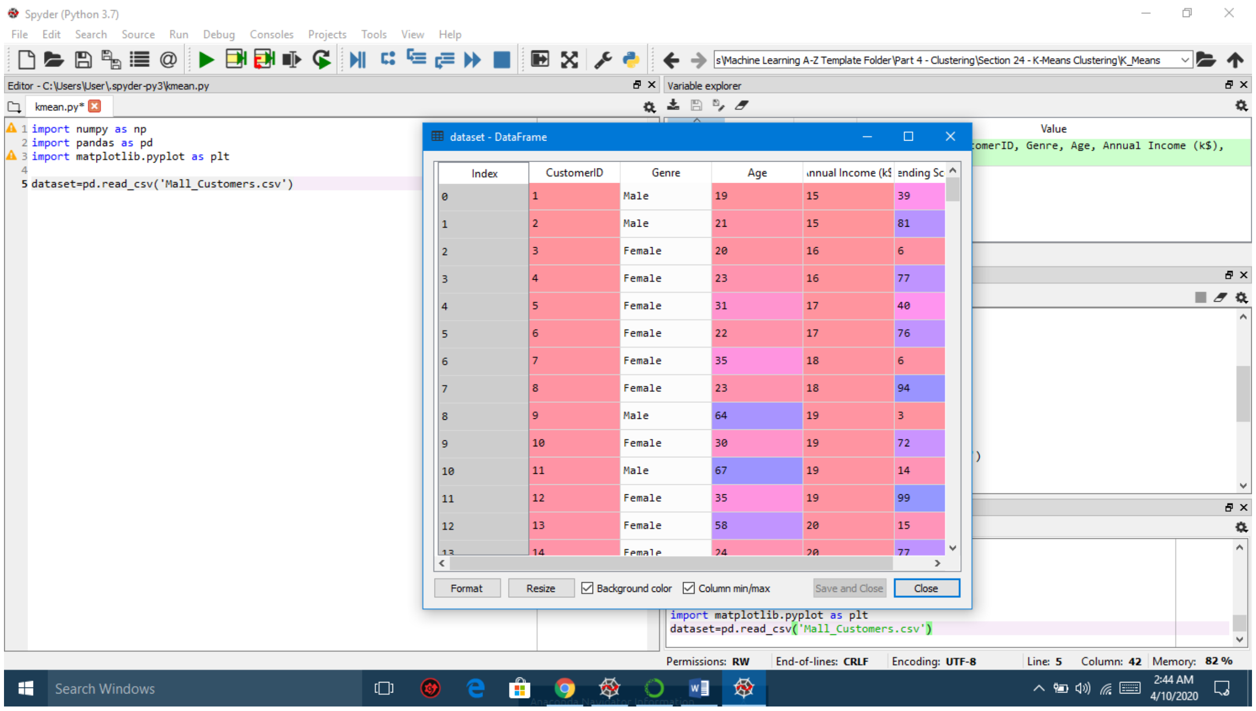
Step 3: Assign each data point to the closet centroid -> forming k cluster.

Step 4: Compute and place new centroid to each cluster.

Step 5: Resign the centroid and goto step 4 else finish.

Code:

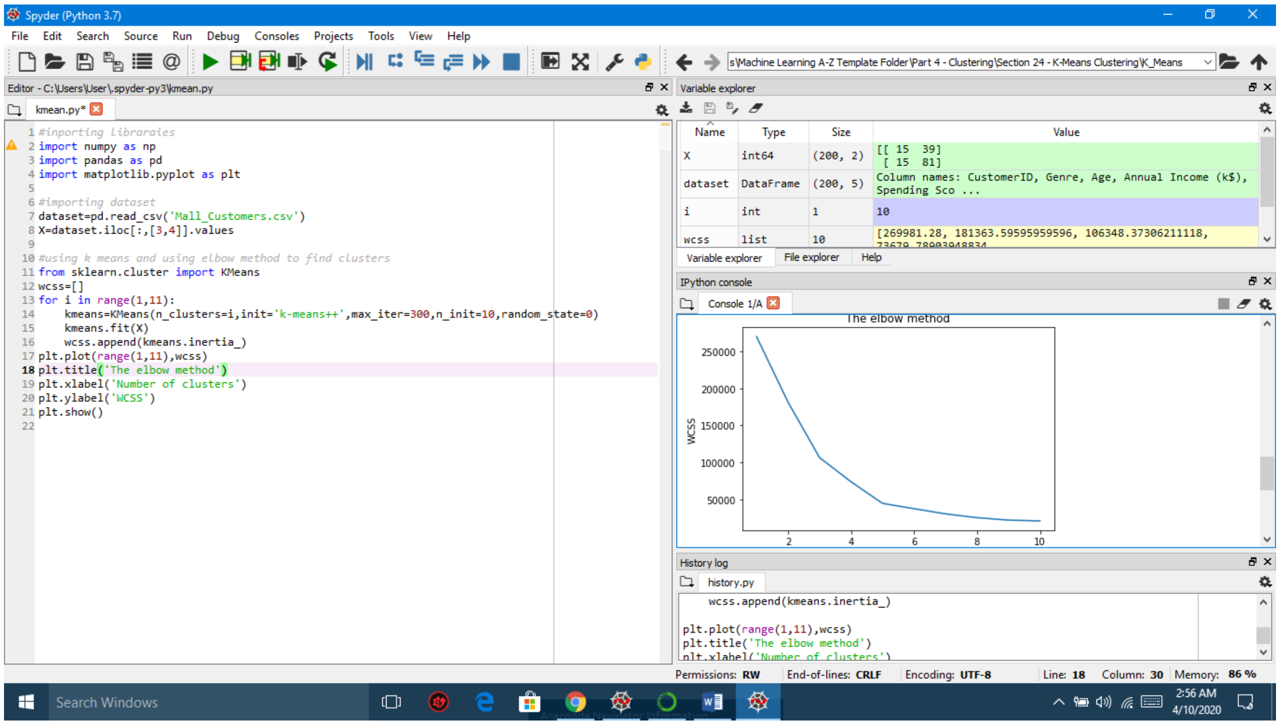
Import all the libraries and get the dataset:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

dataset=pd.read_csv('Mall_Customers.csv')
```

Index	CustomerID	Genre	Age	ınnual Income (k$	ending Sc
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14
11	12	Female	35	19	99
12	13	Female	58	20	15
13	14	Female	24	20	77

Now the elbow method to is been run to find out about the numbers of clusters to optimize


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
s\Machine Learning A-Z Template Folder\Part 4 - Clustering\Section 24 - K-Means Clustering\K_Means
Editor - C:\Users\User\.spyder-py3\kmean.py
kmean.py*
Variable explorer
IPython console
Console 1/A
The elbow method
WCSS
History log
history.py
Search Windows
2:56 AM
4/10/2020

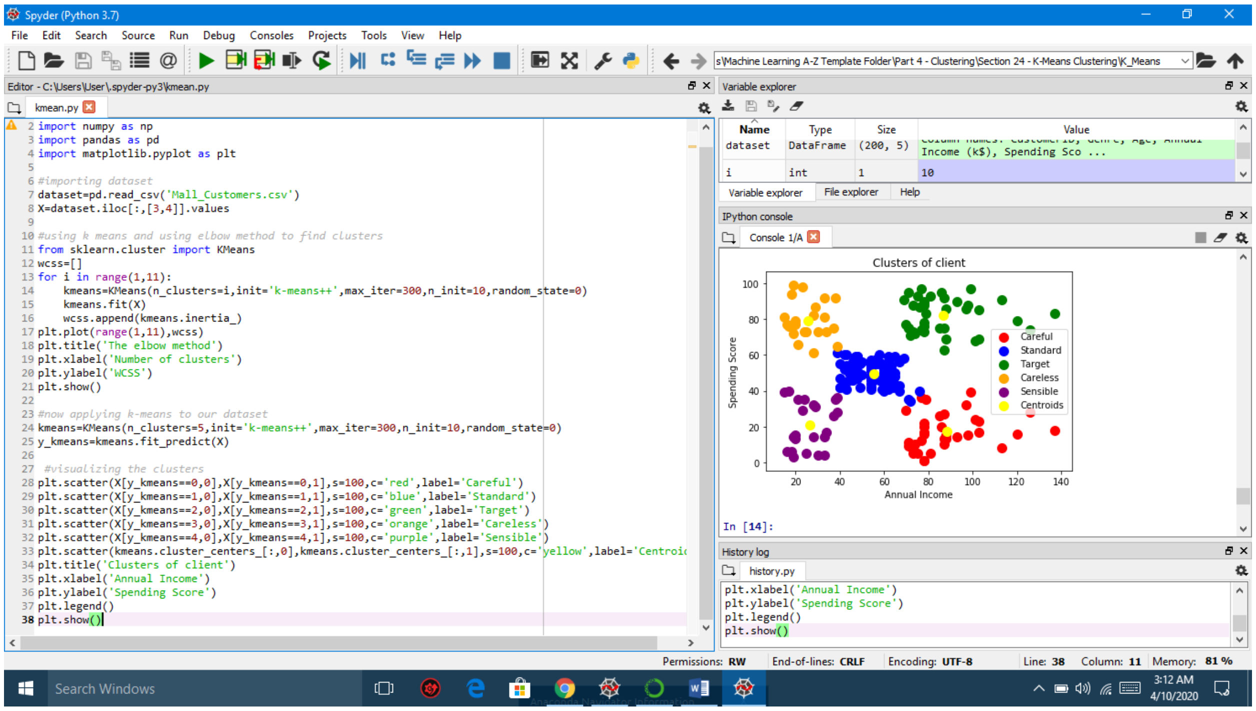
```
#inporting libraraies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#importing dataset
dataset=pd.read_csv('Mall_Customers.csv')
X=dataset.iloc[:,[3,4]].values

#using k means and using elbow method to find clusters
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Name	Type	Size	Value
X	int64	(200, 2)	[[15 39] [15 81]
dataset	DataFrame	(200, 5)	Column names: CustomerID, Genre, Age, Annual Income (k$), Spending Sco ...
i	int	1	10
wcss	list	10	[269981.28, 181363.59595959596, 106348.37306211118,

```
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('The elbow method')
```

```
Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 18   Column: 30   Memory: 86 %
```

Now finally we do have all our clusters which shows the different behaviors of the clients visiting the mall.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#importing dataset
dataset=pd.read_csv('Mall_Customers.csv')
X=dataset.iloc[:,[3,4]].values

#using k means and using elbow method to find clusters
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

#now applying k-means to our dataset
kmeans=KMeans(n_clusters=5,init='k-means++',max_iter=300,n_init=10,random_state=0)
y_kmeans=kmeans.fit_predict(X)

 #visualizing the clusters
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=100,c='red',label='Careful')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=100,c='blue',label='Standard')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=100,c='green',label='Target')
plt.scatter(X[y_kmeans==3,0],X[y_kmeans==3,1],s=100,c='orange',label='Careless')
plt.scatter(X[y_kmeans==4,0],X[y_kmeans==4,1],s=100,c='purple',label='Sensible')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=100,c='yellow',label='Centroic
plt.title('Clusters of client')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.show()
```

Name	Type	Size	Value
dataset	DataFrame	(200, 5)	Income (k$), Spending Sco ...
i	int	1	10

```
In [14]:
```

```
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.show()
```

```
Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 38   Column: 11   Memory: 81 %
```

This is a sample example of how machine learning algorithm can be used to make and insight in the same way we can have the measure on the live smart contract and run that as sample to test and try its authenticity.

Machine Learning with SQL Server 2017 and later along with BigQuery Ml

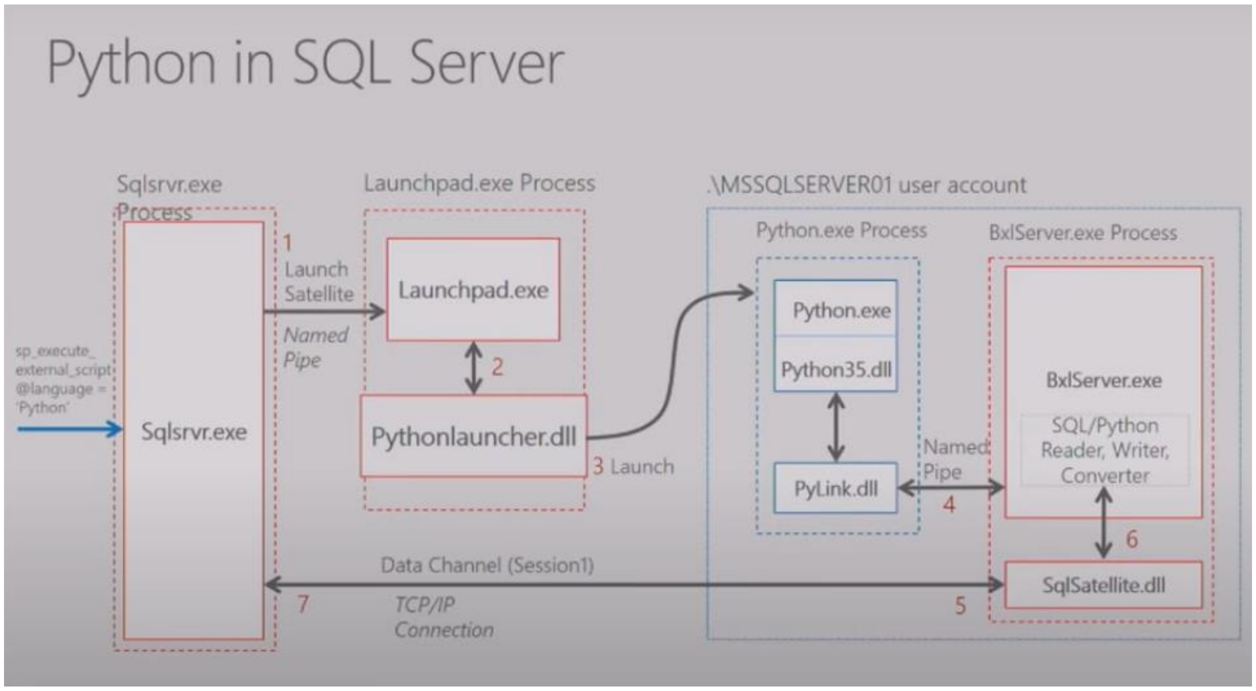
Develop and deploy the machine learning solutions directly in SQL Server.

The biggest Advantage in this scenario is the elimination of Data Movement with-in a database computation as different data scientists uses the data and movement might made some loss as we saw in the excel sheets.

Operationalize Python code and models.

Enterprise Grade Applications are preferred in the safer environment rather than on excel/ access databases.

Regular databases integration as the stored procedure can be called within the application. But, the SQL databases are governed by the database administrators and they must be have a check on how and where the scale must place.



How the execution in code form is done:

Open Sql script.

Then Write,

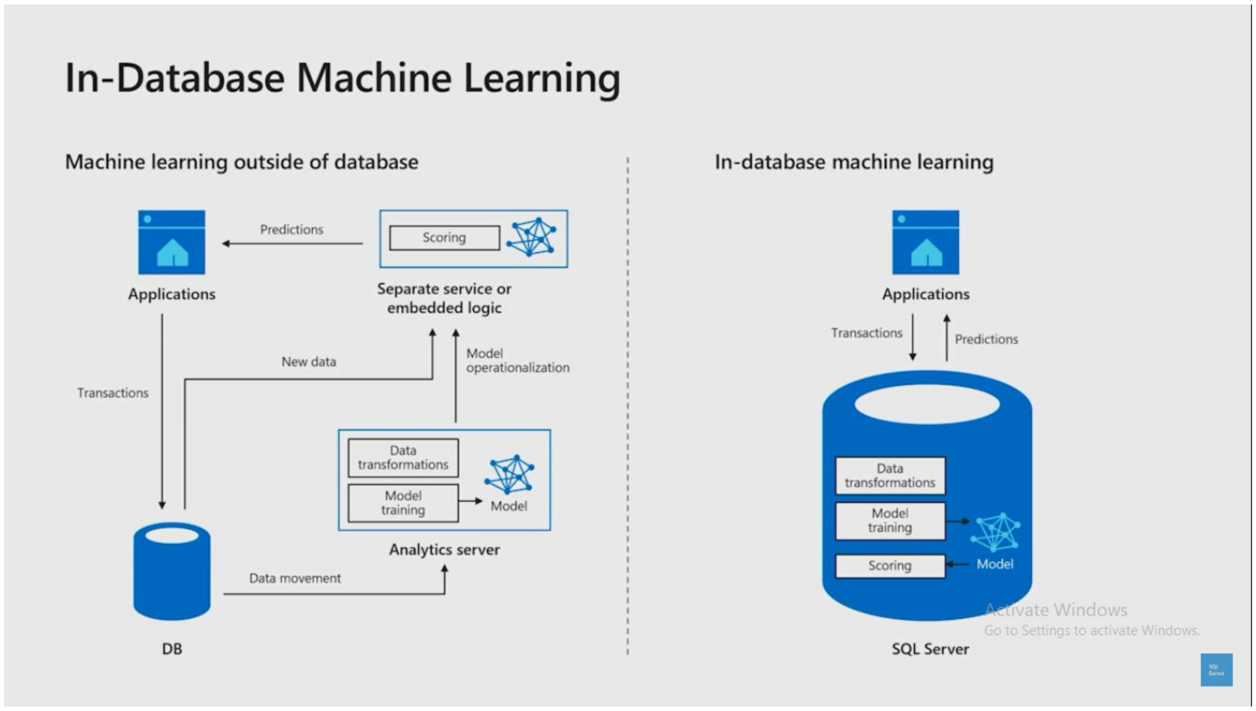
execute sp_execute_externel_script

@language = N'Python',

@script = N'

///write your python code here
'

Also the key advantage can be the parallel execution because the operation in code might start without being completely executing the schema.



MySQL:

It is also one of the open source platform and lightweight used source database:

It is customizable, can have larger datasets and it also supports multiple programming interfaces.

It follows the client/server architecture.

Compatible on many operating systems.

Communication via local connection or internet.

Machine Learning and NOSQL databases:

After the commercial implementation of **Hadoop NoSql** the importance of machine learning now becomes more than ever.

In Rdbms the data is structured while in the Document, Graph form Databases also known as Semi-Structured.

For example let us consider MONGODB:

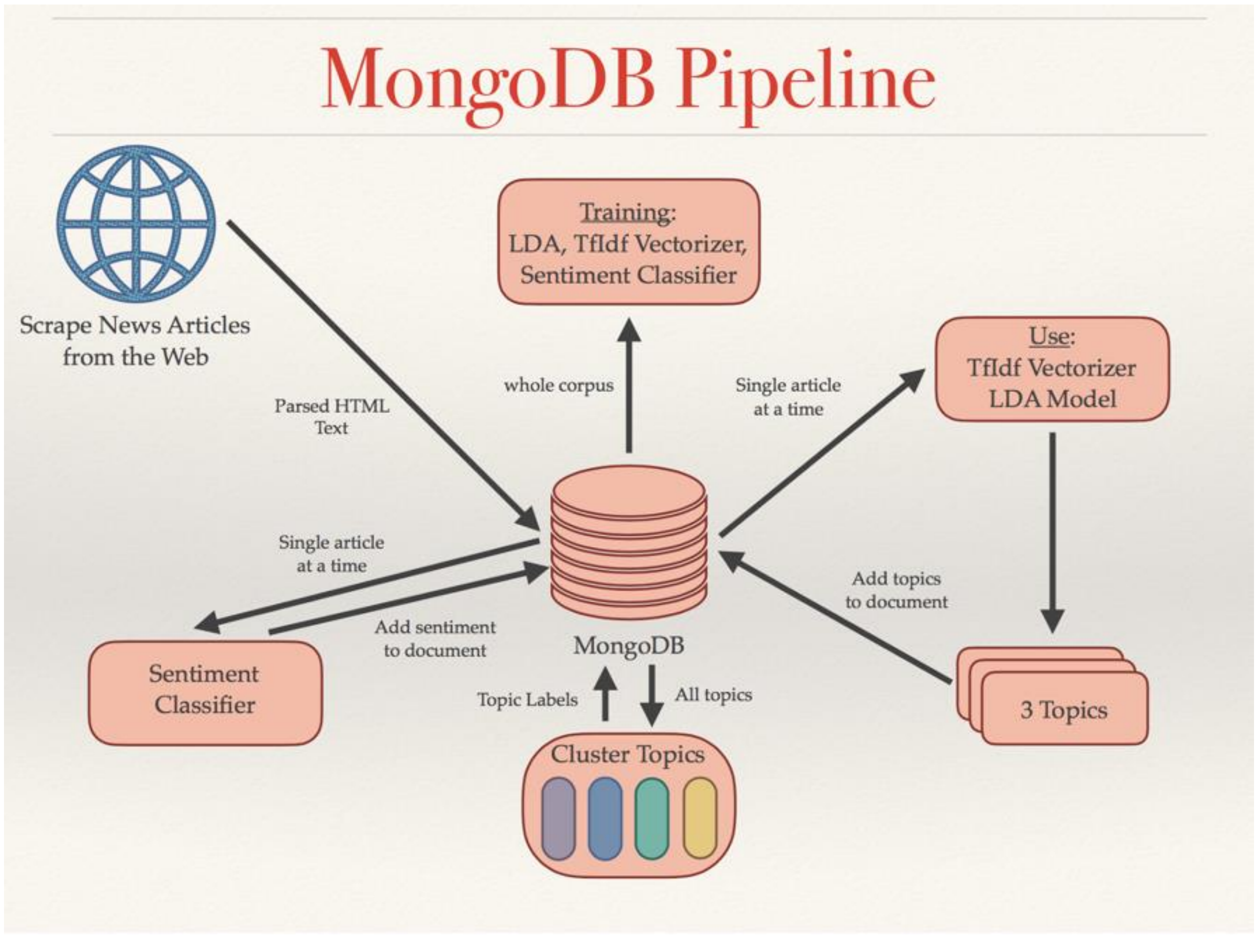
It has flexible data model, indexing and high speed querying than training and executing machine learning algorithms is much faster than the RDBMS.

Advantages:

1) No Complex Joins
2) Schema Less
3) Easy to Scale

Disadvantages:

1) Data Consumption complex due to de-normalization
2) No default transaction supported you have to do it by yourself
3) Map/Reduce is somewhat slower.



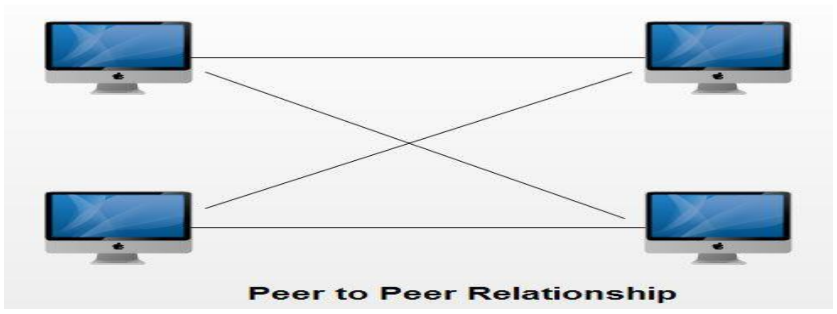
The application of machine learning algorithms that were used in this project:

1) Linear Regression (Simple/Linear)
2) KNN

They will also applied to both the SQL and NoSql in the same way because the working of the algorithm is the same only the conventional medium where the data is taken from is changed.

Peer to Peer Model Training:

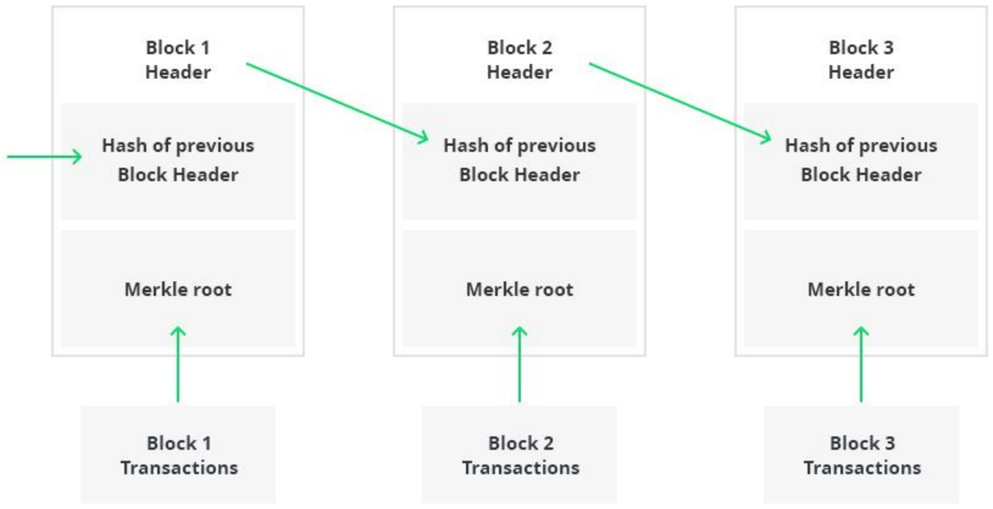
There are memory based learnings which requires huge datasets to train the models. The more instance of the data the better the model will be trained. For this purpose an expensive computational powers are required which makes it quite expensive but the blockchain technology can help make things easier and also the process more scalable. Instead of making the platform on the cloud. There can be the nodes which can be distributed all across the blockchain network. Each Node on the network can contribute in training the model its computational power. The price of each node can be determined according to its work performance.



Construction of AI on Blockchain:

Blockchain is a platform which is immutable storage with high level cryptographic security with other features, though it is not something related to artificial intelligence but it will be a prominent source for AI in the future.

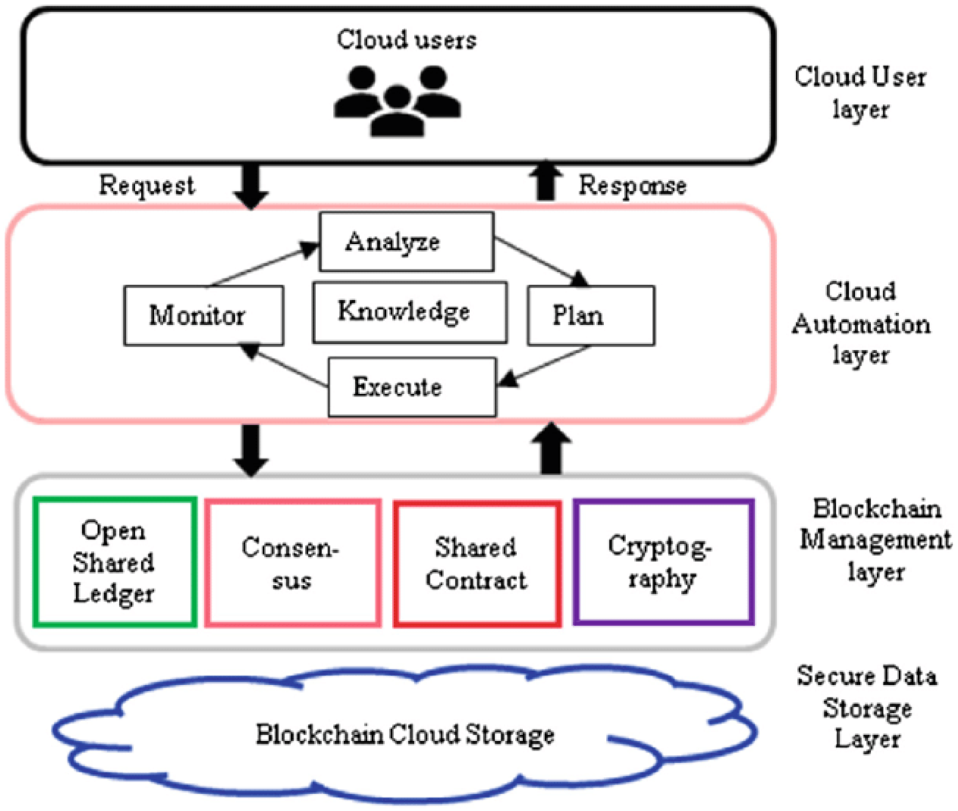
1) Blockchain as a transaction platform:
   Blockchain's first practical implementation was the famous Bitcoin (a cryptocurrency) by a Japanese named Satoshi Nakamoto in 2008 the basic of bitcoin was to have a transaction which was recorded without the involvement of the third party. They are related to the pervious block which are there in the Merkle root a hash function which is the header of each block.



2) Blockchain as a cloud platform:
   As block store data byte by byte, therefore their space is premium and also their each transactions must be verified there must be quite challenging constraints to save them from imposing penalties for efficient calculations while the computational power it brings allows a lot of potential to apply some high computing genetic algorithms which otherwise seems difficult.

The cloud based blockchain platform comprises of four layers.

1) Cloud layer.
2) Cloud Automation layer.
3) Blockchain Management Layer
4) Secure Data Storage Layer



Writing some algorithm which is related to the blockchain is the proof of work algorithm and then the python code which is just an estimated way (not working) thing in written in the intelliJ.

The consensus algorithms in blockchain are:

1) Proof of work.
2) Proof of stake.
3) Proof of burn

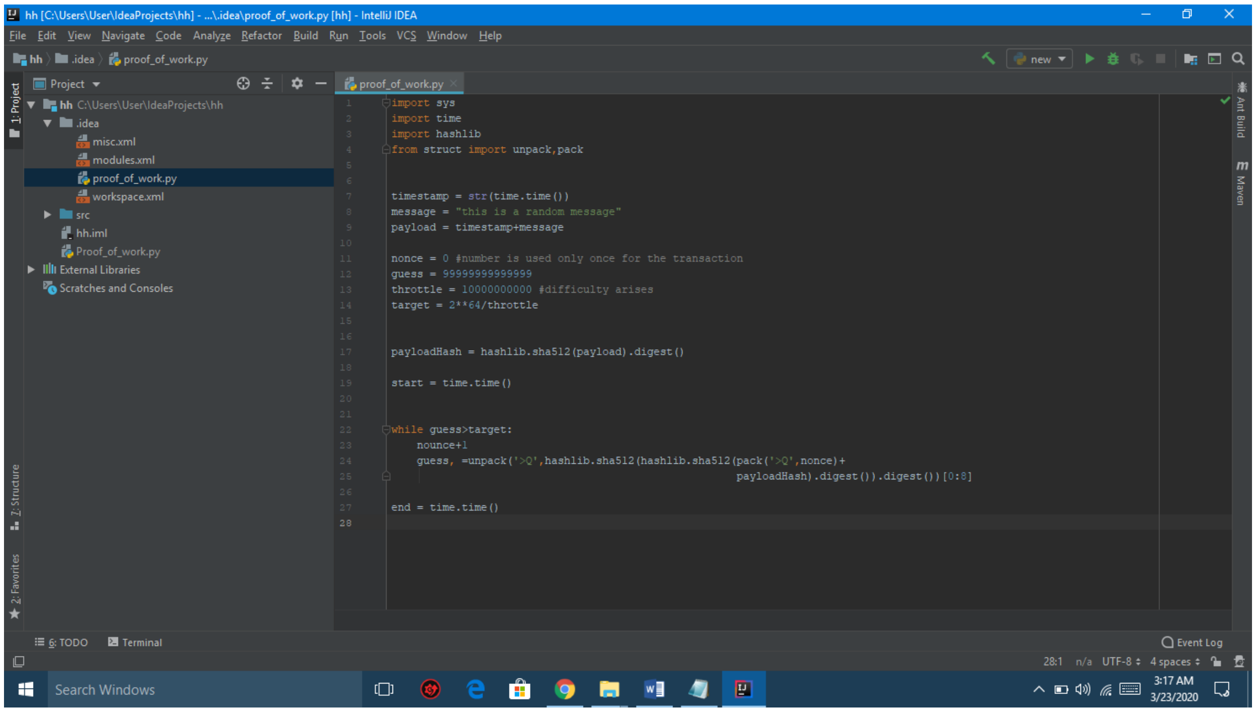
Here is given the example of Proof of work.

Proof of work is a consensus algorithm in blockchain which works in the following manner:

1) Retrieve transactions from the pending.

2) Process work to generate proof.
3) Broadcast proof of work to all the corresponding nodes.
4) Write transactions to the blockchain if proof of work is valid for majority of nodes.

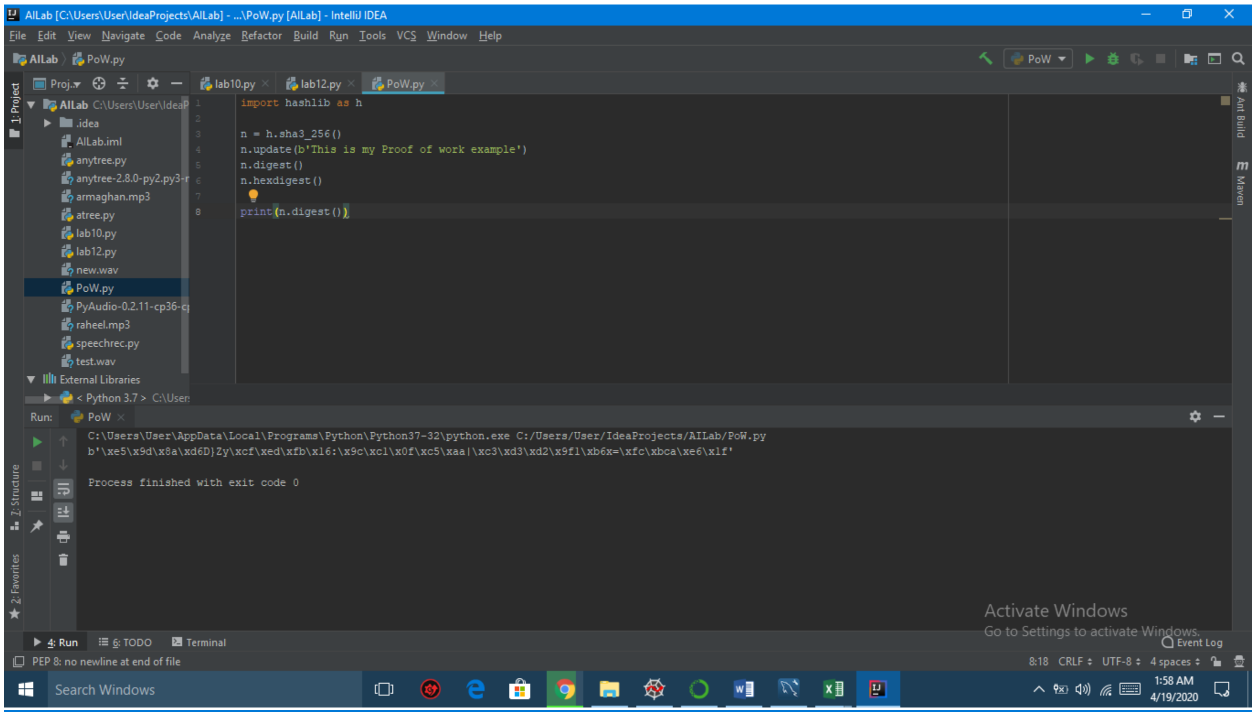
Code:



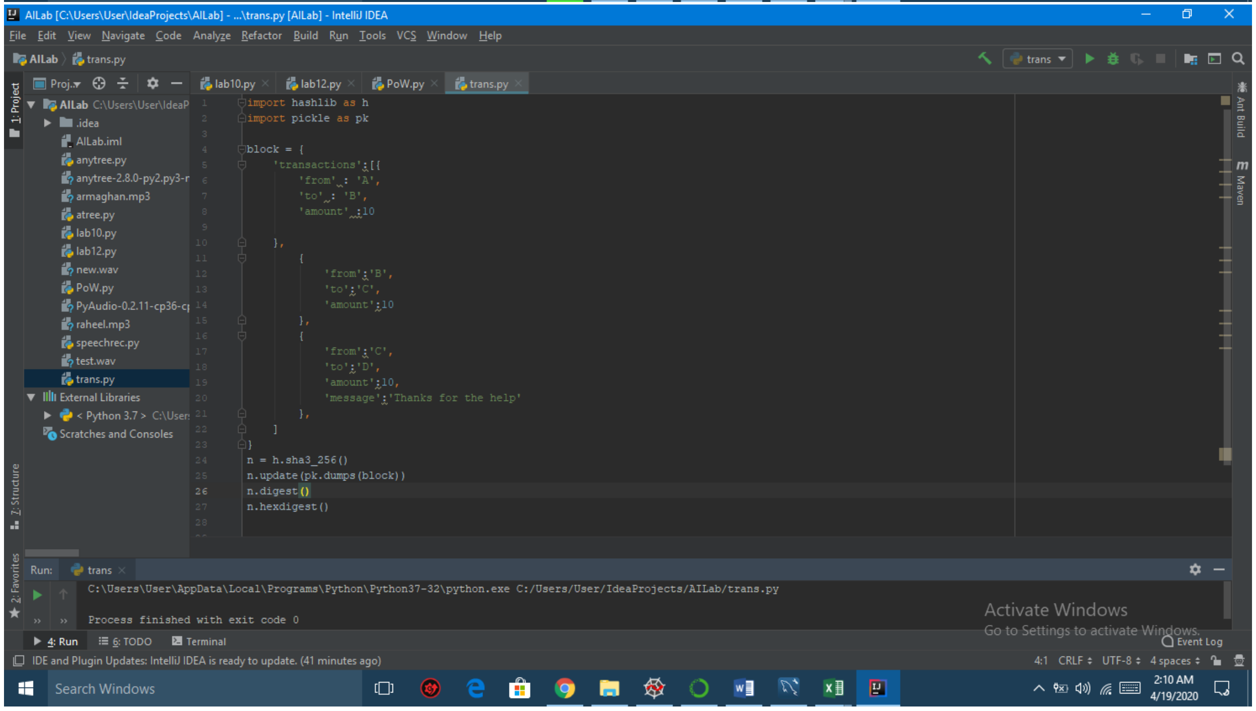
```python
import sys
import time
import hashlib
from struct import unpack,pack


timestamp = str(time.time())
message = "this is a random message"
payload = timestamp+message

nonce = 0 #number is used only once for the transaction
guess = 99999999999999
throttle = 10000000000 #difficulty arises
target = 2**64/throttle


payloadHash = hashlib.sha512(payload).digest()

start = time.time()


while guess>target:
    nounce+1
    guess, =unpack('>Q',hashlib.sha512(hashlib.sha512(pack('>Q',nonce)+
                                                           payloadHash).digest()).digest())[0:8]

end = time.time()
```

So in this code proof of work is done for as long as 10 guesses are made to let the transaction continue.

Another Proof of work along with the transaction block has been constructed which is somewhat similar to how a document in MongoDb is written by scaling that machine learning can be implemented on these data interchangeable format.


```
import hashlib as h

n = h.sha3_256()
n.update(b'This is my Proof of work example')
n.digest()
n.hexdigest()

print(n.digest())
```

```
C:\Users\User\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/User/IdeaProjects/AILab/PoW.py
b'\xe5\x9d\x8a\xd6D}Zy\xcf\xed\xfb\x16:\x9c\xc1\x0f\xc5\xaa|\xc3\xd3\xd2\x9f1\xb6x=\xfc\xbca\xe6\x1f'

Process finished with exit code 0
```


```
import hashlib as h
import pickle as pk

block = {
    'transactions':[{
        'from' : 'A',
        'to' : 'B',
        'amount' :10

    },
        {
            'from':'B',
            'to':'C',
            'amount':10
        },
        {
            'from':'C',
            'to':'D',
            'amount':10,
            'message':'Thanks for the help'
        },
    ]
}
n = h.sha3_256()
n.update(pk.dumps(block))
n.digest()
n.hexdigest()
```

```
C:\Users\User\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/User/IdeaProjects/AILab/trans.py

Process finished with exit code 0
```

Genetic Algorithms in machine learning:

Genetic Algorithms is derived from biology where we have all the possible values which are there in biology right from genes to chromosome and there is a process in which this algorithm works right to have the best combination of solution from the series of solutions. So, starting from the initial population, calculate fitness, selection, crossover, mutation etc.

The cellular Automaton as the neuron of Genetic Algorithm:

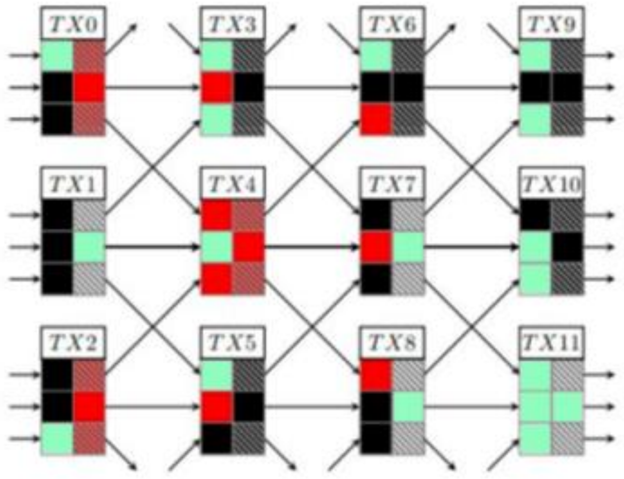
The cellular automaton is the smallest, tightest and general atomic computational unit to act as neuron within genetic algorithm as it makes it particularly easy for blockchain and its very small and tiny transaction side.

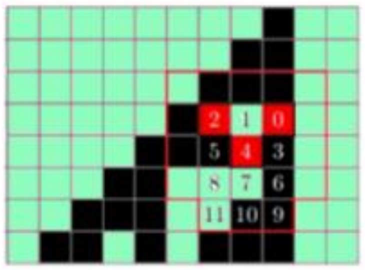
Implementing GA's on BlockChain:

The will be two ways of discussing the implementation of GA in blockchain one is theoretical and the other is by implementation.

While transactions do run in the loop but each transaction must pass the turing test in order to be put forwarded to the cellular automaton. With some rules and validation the code scripts ensure some new transactions with small chance of mutation and while its evolution is simulated. As, iterations in the GA are computationally heavy so they require high intensified which require a lot of resource.

Recursive blockchain with multiple transactions can be achieved instead of single transaction in a single block. This method involves a rule 110 cellular Automaton, CA110 keeps the check bound on the coming iterations with all keeping the track of the previous.





While AI centralization was taken into the consideration for that models were also trained on the Ethereum Blockchain. But, because of the high cost of Ethereum it could have been possible only for the small inputs such as text.

Deep chains: combination of blockchain and deep learning with a self-operating computation graphs with the probabilistic guesses about reality states of world

What we are running on Networks:

1) Information
2) Finances
3) Artificial Intelligence

Intelligence based into smart network

Blockchain will help in creating the platform related to the authenticity and smart network.

Deep learning algorithms for making the predictive analysis

Deep Learning chains Applications:

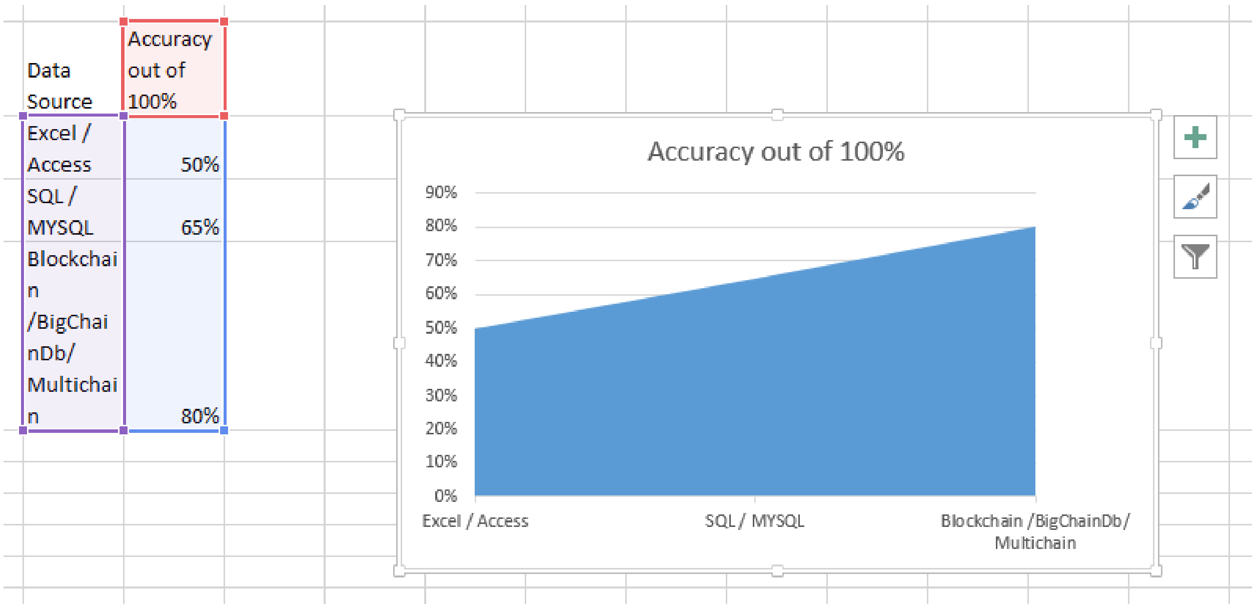
1) Autonomous self-driving, drone delivery (while blockchain helps into keeping the operations secure. Reliable and transformable).
2) Big Health Data (to yield results using predictive analysis to prevent disease while securing medical data with blockchain)
3) Digital Payments (e wallets with the secure and reliable sources but analyzing the fraud predictions of a service agency)

Network Paradigms:

1) Combined
2) Decentralized
3) Distributed

Conclusion:

Now I have drawn the conclusion basis of all three aspects of data source we have seen in this project the excel started with a lot of data preprocessing and inconsistency, where the SQL / MYSQL have solved some of the problems that we faced in excel like easy querying and inconsistency was also reduced, but the database administration having all the control was also one of the main factors, now coming onto the block chain theme which is completely decentralized and peer to peer sort's the problem of data being inconsistent and also it being decentralized the entropy can also be reduced with no administrator needed. Every format has its pros and cons but the project's prime focus was on the data consistency. So, on all the practical and readings I have done I conclude in the graph below.



Data Source	Accuracy out of 100%
Excel / Access	50%
SQL / MYSQL	65%
Blockchain /BigChainDb/ Multichain	80%

Learning and Future Work:

The foremost learning from this project is how important the data is for Artificial Intelligence techniques and this accuracy and consistency is the base of how well the algorithm perform results and the work can not only be in the field of predictions, medical , finance , robotics , industrial automation and several other fields. I would like to take it forward and in fyp if things goes fine with proper scenarios if things went fine. Because database versioning will play a crucial role in further advancement as we have seen how Big Data break in when its implementation had been done. Well, that's all from my side in this artificial Intelligence course project.

References:

1) www.bigchaindb.com

2) 9984 summit Dr Trent Mcconaghy (Co-Founder BigchainDb) Talk

3) Blockchain for Artificial Intelligence

Avi Garg1, Arpit Garg2 1, 2 Student, Dept. of Computer Engineering, Jecrc University, Jaipur, India

4) Artificial Intelligence Implementations on the Blockchain. Use Cases and Future Applications

Konstantinos Sgantzos 1,* and Ian Grigg 2