**Machine Learning**

Machine Learning is a set of techniques to exact knowledge from available data and use that knowledge to make decisions. There is two types of machine learning

1. Deep Learning
2. Mathematical Models

**ML real life examples:**

* Recommendation engines
* Self-Driving cars
* Gamifies Learning and Examples (Ex:Duolingo)
* E-commerce Websites
* Medical Diagnosis
* Getting Right Answers(Quora)
* Spam filters

**ML Techniques**

**Linear Progression Single Variable**

* This method is only suitable for data linear or close to linear
* **Example Dataset**

|  |  |
| --- | --- |
| Area | Price |
| 2600 | 550000 |
| 3000 | 565000 |
| 3200 | 610000 |
| 3600 | 680000 |
| 4000 | 725000 |

* **The code and steps to train and get new output:**

\* This program shows the function as y=m\*x+c

**Step 1:Importing**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import linear\_model

**Taking the file to Python**

df = pd.read\_filetype("filename.filetype")

df

**Plotting**

%matplotlib inline

plt.scatter(df.Area , df.Price ,color='red' ,marker='\*')

plt.xlim(2500 , 4500)

plt.xlabel('Area [sq ft]')

plt.ylabel('Price[Taka]')

plt.show()

**Curvefitting**

reg = linear\_model.LinearRegression()

reg.fit(df[['Area']],df.Price)

**Output Code**

reg.predict([[3300]])

**Gotten Output**

array([628715.75342466])

**Coefficient(m)**

Code: reg.coef\_

Result: array([135.78767123])

**Constant(c)**

Code: reg.intercept\_

Result: np.float64(180616.43835616432)

**Linear Progression Multiple Variables**

* Here, the result depends on more than one variable
* **Example Dataset**

|  |  |  |  |
| --- | --- | --- | --- |
| **Area[sq ft]** | **Bedrooms** | **Age** | **Price** |
| 2600 | 3 | 20 | 550000 |
| 3000 | 4 | 15 | 565000 |
| 3200 |  | 18 | 610000 |
| 3600 | 3 | 30 | 595000 |
| 4000 | 5 | 8 | 760000 |

**\*Find the price of**

**3000 sq ft,3 bedrooms,40 years old**

**2500 sq ft,4 bedrooms,5 years old**

* Data format: y=m1\*x1+m2\*x2+m3\*x3+……+mnxn+b
* To handle missing data:

Code: (Taking Median)

i) import math

median\_Bedrooms = math.floor(df.Bedrooms.median())

median\_Bedrooms

ii) df.Bedrooms = df.Bedrooms.fillna(median\_Bedrooms)

df

* To train the machine

Code:

reg = linear\_model.LinearRegression()

reg.fit(df[["Area","Bedrooms","Age"]],df.Price)

* To Predict the questioning result:

**Code:**

reg.predict([[3000,3,40]])

reg.predict([[2500,4,5]])

**Result:**

array([444400.])

array([588625.])

**Gradient Descant and Cost Function**

From the example dataset of LPSV, it can be shown that data is scattered.  
**So, how can we find the best fitting line?**

One way is to draw any random line and find the difference().

And find Mean Squared Error

m.s.e is also known as cost function

**Gradient descent is an algorithm that finds the best fit line for given data set.**

We can find the minimum cost by derivation and finding the minimum.

**Saving Model**

**In Pickle**

Code:

import pickle

with open ('model\_pickle' ,'wb') as f:

pickle.dump(reg,f)

(The file is saved now)

***To run this file:***

Code:

with open ('model\_pickle' ,'rb') as f: # reading the model

mp=pickle.load(f)

(Now, the model can be used for prediction)

**Using Joblib (Sklearn)**

\*The Difference between Joblib and Pickle is that Joblib is more efficient for larger data

Code:

import joblib

joblib.dump(reg,'model\_joblib')

(The file is saved now)

***To run this file:***

mj=joblib.load('model\_joblib')

(Now, the model can be used for prediction)

**Dummy Variables and One hot encoding**

Example Dataset:

|  |  |  |
| --- | --- | --- |
| Town | Area | Price |
| Dhaka | 2600 | 550000 |
| Dhaka | 3000 | 565000 |
| Dhaka | 3200 | 610000 |
| Dhaka | 3600 | 680000 |
| Dhaka | 4000 | 725000 |
| Rajshahi | 2600 | 585000 |
| Rajshahi | 2800 | 615000 |
| Rajshahi | 3300 | 650000 |
| Rajshahi | 3600 | 710000 |
| Khulna | 2600 | 575000 |
| Khulna | 2900 | 600000 |
| Khulna | 3100 | 620000 |
| Khulna | 3600 | 695000 |

Questions:

Build a predictor function to predict price of a home

1. With 3400 sq ft area in Rajshahi
2. 2800 sq ft home in Khulna

\*The city variable of the above table is known as the nominal variables

As machines cannot deal with the words(nominal variables), we need to turn the words(City names in this case) to a number. To do this, the following table can be used:

|  |  |  |
| --- | --- | --- |
| Dhaka | Rajshahi | Khulna |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |

**This is known as One Hot Encoding and The Variable Here is known as Dummy Variables**

Code to get dummy file:

dummies=pd.get\_dummies(df.Town).astype(int)

By Label Encoder:

le = LabelEncoder()

**Spilt dataset into Training and Testing**

Example Datasheet:

|  |  |  |
| --- | --- | --- |
| Mileage | Age | Price |
| 69000 | 6 | 18000 |
| 35000 | 3 | 34000 |
| 57000 | 5 | 26100 |
| 22500 | 2 | 40000 |
| 46000 | 4 | 31500 |
| 59000 | 5 | 26750 |
| 52000 | 5 | 32000 |
| 72000 | 6 | 19300 |
| 91000 | 8 | 12000 |
| 67000 | 6 | 22000 |
| 83000 | 7 | 18700 |
| 79000 | 7 | 19500 |
| 59000 | 5 | 26000 |
| 58780 | 4 | 27500 |
| 82450 | 7 | 19400 |
| 25400 | 5 | 35000 |
| 28000 | 2 | 35500 |
| 69000 | 5 | 19700 |
| 82600 | 8 | 12800 |
| 52000 | 5 | 28200 |

Code:

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=10)

Logistic Regression

\*Linear Regression can be used where The predicted value is continuous such as predicting Home Prices, Weather, Stock Prices etc.

But, while the predicted value is categorical(like a email is spamming or not, if a customer buy a thing or not etc.), it is called classification problem. To solve classification problems, Logistic Regression is used.

**Binary Classification**

Binary Classification has only two answers, yes or no.

Example Dataset;(1=yes , 0=no)

|  |  |
| --- | --- |
| age | Insurance |
| 22 | 0 |
| 25 | 0 |
| 47 | 1 |
| 52 | 0 |
| 46 | 1 |
| 56 | 1 |
| 55 | 0 |
| 60 | 1 |
| 62 | 1 |
| 61 | 1 |
| 18 | 0 |
| 28 | 0 |
| 27 | 0 |
| 29 | 0 |
| 49 | 1 |
| 55 | 1 |
| 25 | 1 |
| 58 | 1 |
| 19 | 0 |
| 18 | 0 |
| 21 | 0 |
| 26 | 0 |
| 40 | 1 |
| 45 | 1 |
| 50 | 1 |
| 54 | 1 |
| 23 | 0 |

Code:

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

**Multiclass Classification**

\*Its coding is same as the binary classification. The only difference is that the possible answer here is more than two.

Decision Tree

Example Datasheet: ;(1=yes , 0=no)

|  |  |  |  |
| --- | --- | --- | --- |
| Company | Job | Degree | Salary\_more\_than 100k |
| google | sales executive | Bachelors | 0 |
| google | sales executive | Masters | 0 |
| google | business manager | Bachelors | 1 |
| google | business manager | Masters | 1 |
| google | computer programmer | Bachelors | 0 |
| google | computer programmer | Masters | 1 |
| abc pharma | sales executive | Masters | 0 |
| abc pharma | computer programmer | Bachelors | 0 |
| abc pharma | business manager | Bachelors | 0 |
| abc pharma | business manager | Masters | 1 |
| facebook | sales executive | Bachelors | 1 |
| facebook | sales executive | Masters | 1 |
| facebook | business manager | Bachelors | 1 |
| facebook | business manager | Masters | 1 |
| facebook | computer programmer | Bachelors | 1 |
| facebook | computer programmer | Masters | 1 |

We should always use approach where we have High Information Gain (Company in this case as it has a lower entropy)

Code:

from sklearn import tree

model = tree.DecisionTreeClassifier()

Support Vector Machine

Support vector machine draws a hyper plane in n dimensional space such that it maximizes margin between classification groups.

(Two Pictures of (High Gamma, Low Gamma) & (High Regularization, Low Regularization) would be added here)

Kernel means to create a transformation on existing features so that we can draw the decision boundary easily.  
Code:

from sklearn.svm import SVC

model = SVC

Random Forest

Code:

Polynomial Regression

Polynomial Regression is used where the data is non-linear. General Polynomial Equation:

This is an n degree polynomial.