**# # Polynomial Regression**

# - Build a model which can predict the salary of an employee based on his/her Level(Grade). So we can guess whether a particular candidate is telling Truth/about his/her salary based on the Level provided by him.

#

**# - Input Data : Level and Salary**

**# - Types of Regression**

# 1. Simple LR Y = a + bX

# 2. Multiple LR Y = a + bX1 + cX2 + dX3..

# 3. Polynomial LR Y = a + bX + cX^2 +...

#

# In[315]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# In[316]:

**# data collection**

dataset = pd.read\_csv('./Datasets/Position\_Salaries.csv')

dataset.head()

# In[317]:

**# separate input and output**

X = dataset.iloc[:,1:2].values

Y = dataset.iloc[:,2].values

X.shape

# In[318]:

# data Analysis

plt.scatter(X,Y)

plt.xlabel('Level')

plt.ylabel('Salary')

plt.show()

# In[319]:

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

# Fitting /Training the model

lin\_reg.fit(X,Y)

predicted\_value = lin\_reg.predict(X)

**# Visualizing the Linear Regression results**

plt.scatter(X,Y, color='red')

plt.plot(X, predicted\_value, color='blue')

plt.title("Truth or Bluff( Linear Regression)")

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

**# ### Fitting Polynomial Regression to the dataset**

# In[320]:

from sklearn.preprocessing import PolynomialFeatures

# try with different values of equation degree also like 2 or 3 or 5

poly\_reg = PolynomialFeatures(degree=4)

X\_poly = poly\_reg.fit\_transform(X)

X\_poly

# In[321]:

lin\_reg\_2 = LinearRegression()

lin\_reg\_2.fit(X\_poly, Y)

predicted\_value = lin\_reg\_2.predict(poly\_reg.fit\_transform(X))

**# visualising the Poly Reg results**

plt.scatter(X,Y, color='red')

plt.plot(X, predicted\_value, color='blue')

plt.title("Truth or Bluff( Polynomial Regression)")

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

**# # Decision Tree Classifier**

# In[322]:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn import metrics

from sklearn import tree

# In[323]:

**# data collection/loading**

data = pd.read\_csv('./Datasets/pacific.csv')

data.head()

# In[324]:

**# Data Wrangling**

pred\_columns = data[:]

pred\_columns.drop(['ID', 'Name', 'Date', 'Time', 'Event', 'Status', 'Latitude','Longitude'],

inplace=True,

axis=1)

pred\_columns.head()

# In[325]:

**# list of input columns**

input\_columns = list(pred\_columns.columns)

[input\_columns]

# In[326]:

sns.countplot(data['Status'], label='count')

plt.show()

# In[327]:

**# splitting the data into training and testing**

train, test = train\_test\_split(data, test\_size=0.3)

train.head()

# In[328]:

test.head()

# In[329]:

**# relevant columns for training**

train\_X = train[input\_columns]

train\_Y = train['Status']

**# relevant columns for testing**

test\_X = test[input\_columns]

test\_Y = test['Status']

# In[330]:

**# model**

model = tree.DecisionTreeClassifier()

**# training the model**

model.fit(train\_X, train\_Y)

**# testing the model**

prediction = model.predict(test\_X)

**# accuracy measurement for the model**

metrics.accuracy\_score(test\_Y, prediction)

# In[331]:

#pd.DataFrame(prediction, test\_Y)

# In[332]:

tree.export\_graphviz(model, out\_file='decision\_tree.dot', feature\_names=input\_columns)

# paste the binary file O/P in http://www.webgraphviz.com/

**# # Random Forest**

**# - ensemble of many decision trees**

# In[333]:

from sklearn.ensemble import RandomForestClassifier

# number of decision tree is 100

model = RandomForestClassifier(n\_estimators=100)

**#training**

model.fit(train\_X, train\_Y)

**# testing**

prediction = model.predict(test\_X)

**# accuracy**

metrics.accuracy\_score(test\_Y, prediction)