Machine Learning Modeling

1. Regression
2. Classification

import libs

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

import dataset

```
# from google.colab import files
# uploaded = files.upload()
# D5data3.csv

import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

```
dataset = pd.read_csv('D5data3.csv')
dataset.head()
```

	Age	Height
0	10	138
1	11	138
2	12	138
3	13	139
4	14	139

```
dataset.shape
```

```
(71, 2)
```

```
dataset.describe()
```

	Age	Height
count	71.000000	71.000000
mean	45.000000	160.873239
std	20.639767	20.842902
min	10.000000	138.000000
25%	27.500000	143.500000
50%	45.000000	155.000000
75%	62.500000	171.500000
max	80.000000	208.000000

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71 entries, 0 to 70
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Age     71 non-null     int64
 1   Height  71 non-null     int64
dtypes: int64(2)
memory usage: 1.2 KB
```

identify X & Y

```
# independent vars
x = dataset.iloc[ : , :-1].values
x[:5]
```

```
array([[10],
       [11],
       [12],
       [13],
       [14]], dtype=int64)
```

```
# dependent vars
y = dataset.iloc[ : , 1].values
y[:5]
```

```
array([138, 138, 138, 139, 139], dtype=int64)
```

Splitting

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

```
x_train[:5]
```

```
array([[12],
       [50],
       [21],
       [13],
       [64]], dtype=int64)
```

```
y_train[:5]
```

```
array([138, 160, 141, 139, 173], dtype=int64)
```

Transformation- skipped

- all data is numerical, so transformation is skipped

Linear Regression

Modeling Linear Regression (for comparison)

```
from sklearn.linear_model import LinearRegression
```

```
LinMod = LinearRegression()
```

Training Linear Regression (for comparison)

```
LinMod.fit(x_train, y_train)
```

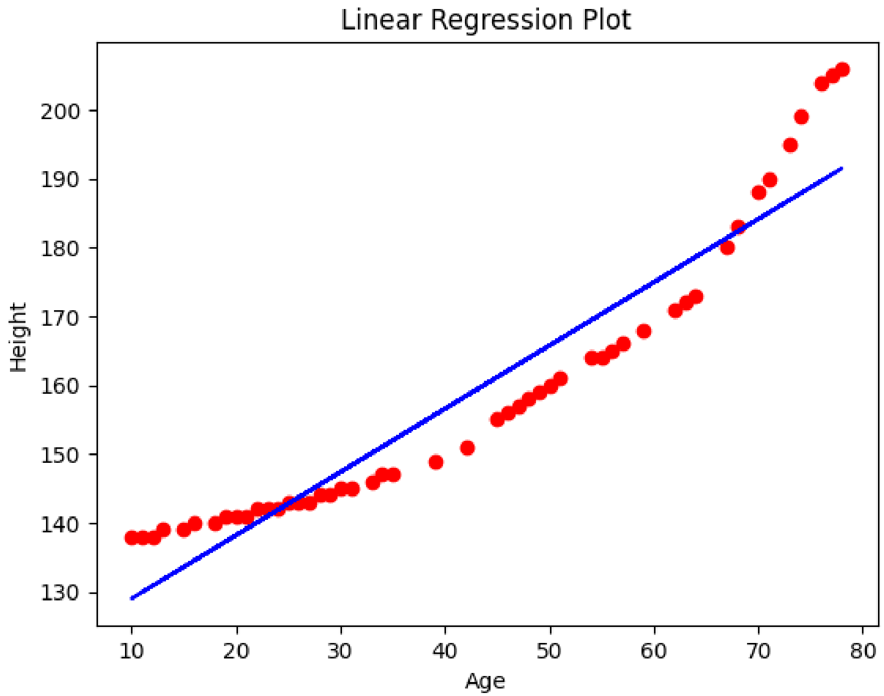
```
▾ LinearRegression
LinearRegression()
```

Visualize Linear Regression (for comparison)

```
import matplotlib.pyplot as plt
```

```
plt.scatter(x_train, y_train, c='red')
plt.plot(x_train, LinMod.predict(x_train), color='blue')
plt.title('Linear Regression Plot')
plt.xlabel('Age')
plt.ylabel('Height')
```

```
Text(0, 0.5, 'Height')
```



Polynomial Regression (degree=2)

Modelling Polynomial Regression (degree=2)

```
from sklearn.preprocessing import PolynomialFeatures
```

```
x_PolyNom = PolynomialFeatures(degree=2)
# degree=2
```

Training Polynomial Regression on `x_train` (degree=2)

```
x_PolyNom = x_PolyNom.fit_transform(x_train)
```

Modeling Linear Regression (degree=2)

```
PolyModel = LinearRegression()
```

Training Linear Regression on PolyNomial Regression (degree=2)

- Fitting Polynomial Regression Model into the Linear Regression Model

```
PolyModel.fit(x_PolyNom, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

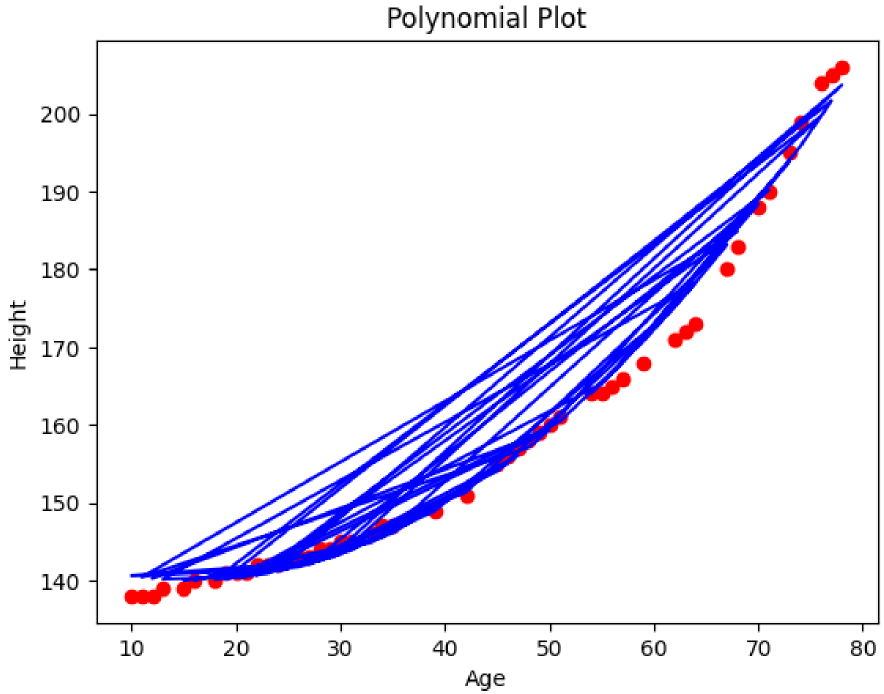
Visualize Polynomial Regression (degree=2)

Training Data vs. Predicted Data (degree=2)

```
plt.scatter(x_train, y_train, c='red')
# Training Data on datapoints
plt.plot(x_train, PolyModel.predict(x_PolyNom), color='blue')
```

```
# Training data vs. Predicted data on line,
# it should overfit as Predictions are on Training Data
plt.title('Polynomial Plot')
plt.xlabel('Age')
plt.ylabel('Height')
```

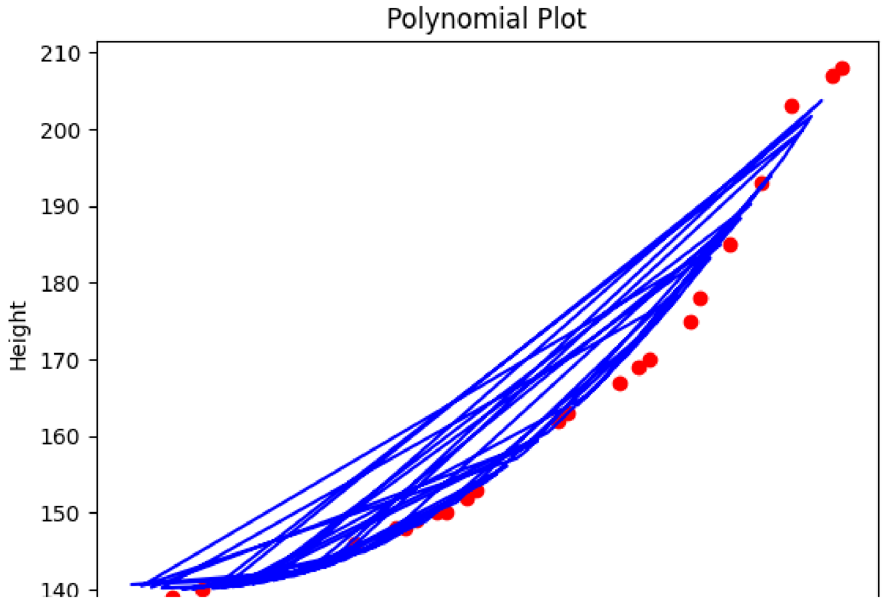
```
Text(0, 0.5, 'Height')
```



Testing Data vs. Predicted Data (degree=2)

```
plt.scatter(x_test, y_test, c='red')
# Testing Data on datapoints
plt.plot(x_train, PolyModel.predict(x_PolyNom), color='blue')
# Training data vs. Predicted data on line
plt.title('Polynomial Plot')
plt.xlabel('Age')
plt.ylabel('Height')
```

```
Text(0, 0.5, 'Height')
```



Polynomial Regression (degree=3)

Modeling Polynomial Regressoin (degree=3)

```
from sklearn.preprocessing import PolynomialFeatures
```

```
PolyNom3 = PolynomialFeatures(degree=3)
# degree=3
```

Training Polynomial Regression on `x_train` (degree=3)

```
x_PolyNom3 = PolyNom3.fit_transform(x_train)
```

Modeling Linear Regression (degreee=3)

```
PolyModel3 = LinearRegression()
```

Training Linear Regression on Polynomial Regression (degree=3)

- Fitting Polynomial Regression Model into the Linear Regression Model

```
PolyModel3.fit(x_PolyNom3, y_train)
```

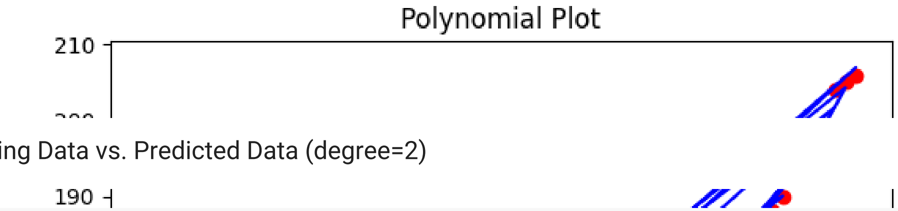
```
▾ LinearRegression
LinearRegression()
```

Visualize Polynomial Regression (degree=3)

Training Data vs. Predicted Data (degree=3)

```
plt.scatter(x_train, y_train, c='red')
# Training Data on datapoints
plt.plot(x_train, PolyModel3.predict(x_PolyNom3), color='blue')
# Training data vs. Predicted data on line,
# it should overfit as Predictions are on Training Data
plt.title('Polynomial Plot')
plt.xlabel('Age')
plt.ylabel('Height')
```

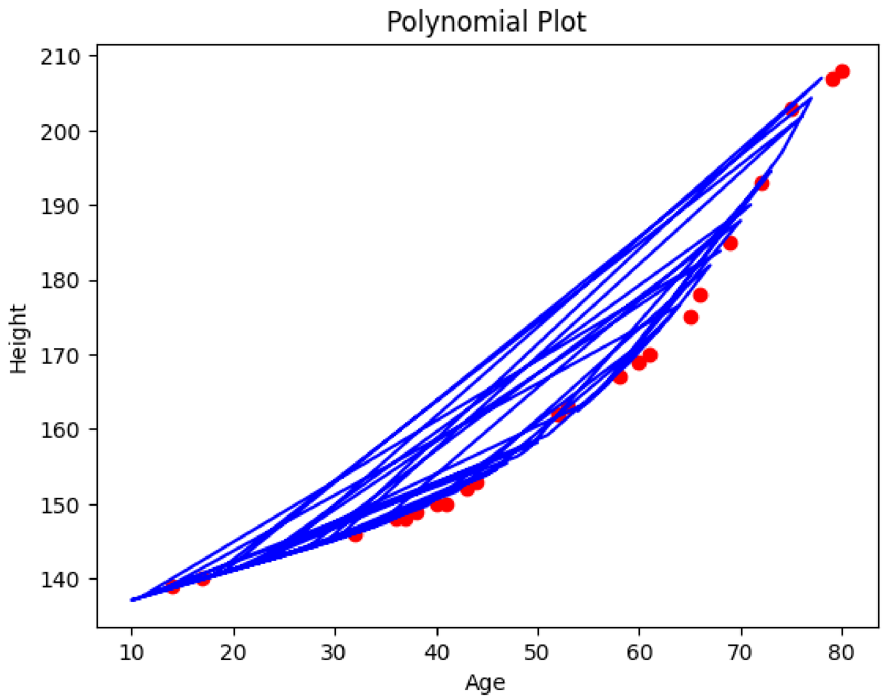
```
Text(0, 0.5, 'Height')
```



Testing Data vs. Predicted Data (degree=2)

```
plt.scatter(x_test, y_test, c='red')
# Testing Data on datapoints
plt.plot(x_train, PolyModel3.predict(x_PolyNom3), color='blue')
# Training data vs. Predicted data on line
plt.title('Polynomial Plot')
plt.xlabel('Age')
plt.ylabel('Height')
```

```
Text(0, 0.5, 'Height')
```



Prediction

Linear Model prediction

```
y_pred = LinMod.predict(x_test)
y_pred[:5]
```

```
array([152.90005813, 153.8194466 , 173.1266044 , 149.22250426,
       156.577612  ])
```

Polynomial Model Prediction (degree=2)

Polynomial Model Prediction (degree=2) with Training Data

```
y_pred_poly_2 = PolyModel.predict(x_PolyNom)
y_pred_poly_2[:5]
```

```
array([140.30766833, 159.29470102, 140.5097378 , 140.1984773 ,
       178.26945458])
```

Polynomial Model Prediction (degree=2) with Testing Data

```
# y_test_pred_poly_2 = PolyModel.predict(PolyModel.fit(x_PolyNom.fit_transform(x_train), y_train))
xt_PolyNom = PolynomialFeatures(degree=2)
xt_PolyNom = xt_PolyNom.fit_transform(x_test)
tPolyModel = LinearRegression()
tPolyModel.fit(xt_PolyNom, y_test)
y_test_pred_poly_2 = tPolyModel.predict(xt_PolyNom)

y_test_pred_poly_2[:5]
```

```
array([146.29385284, 146.93520085, 168.58178925, 144.08249891,
       149.07166774])
```

Polynomial Model Prediction (degree=3)

Polynomial Model Prediction (degree=3) with Training Data

```
y_pred_poly_3 = PolyModel3.predict(x_PolyNom3)
y_pred_poly_3[:5]
```

```
array([138.01406493, 158.21534659, 141.55909917, 138.44053848,
       176.57869783])
```

Polynomial Model Prediction (degree=3) with Testing Data

```
# y_pred_poly_3 = PolyModel3.predict(x_PolyNom3)
xt_PolyNom3 = PolynomialFeatures(degree=3)
xt_PolyNom3 = xt_PolyNom3.fit_transform(x_test)
tPolyModel3 = LinearRegression()
tPolyModel3.fit(xt_PolyNom3, y_test)
y_test_pred_poly_3 = tPolyModel3.predict(xt_PolyNom3)


y_test_pred_poly_3[:5]
```

```
array([147.77118324, 148.3237655 , 167.20114838, 145.76885431,
       150.12603947])
```

Evaluation metric

Evaluation metric for Linear model Prediction

```
from sklearn import metrics
```

```
r_square = metrics.r2_score(y_test, y_pred)
# calculating R-Square for Linear model
r_square
```

```
0.8727873738671587
```

Evaluation metric for Polynomial model Prediction

Evaluation metric for Polynomial model Prediction (degree=2)

```
r_square = metrics.r2_score(y_test, y_test_pred_poly_2)
# calculating R-Square for predictions of Polynomial model with degree=2
r_square
```

```
0.9900964432101119
```

Evaluation metric for Polynomial model Prediction (degree=3)

```
r_square = metrics.r2_score(y_test, y_test_pred_poly_3)
# calculating R-Square for predictions of Polynomial model with degree=3
r_square
```

```
0.9941630782615041
```