Linear Regression

In [1]:

```
#basics libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

IRIS dataset :

In [61]:

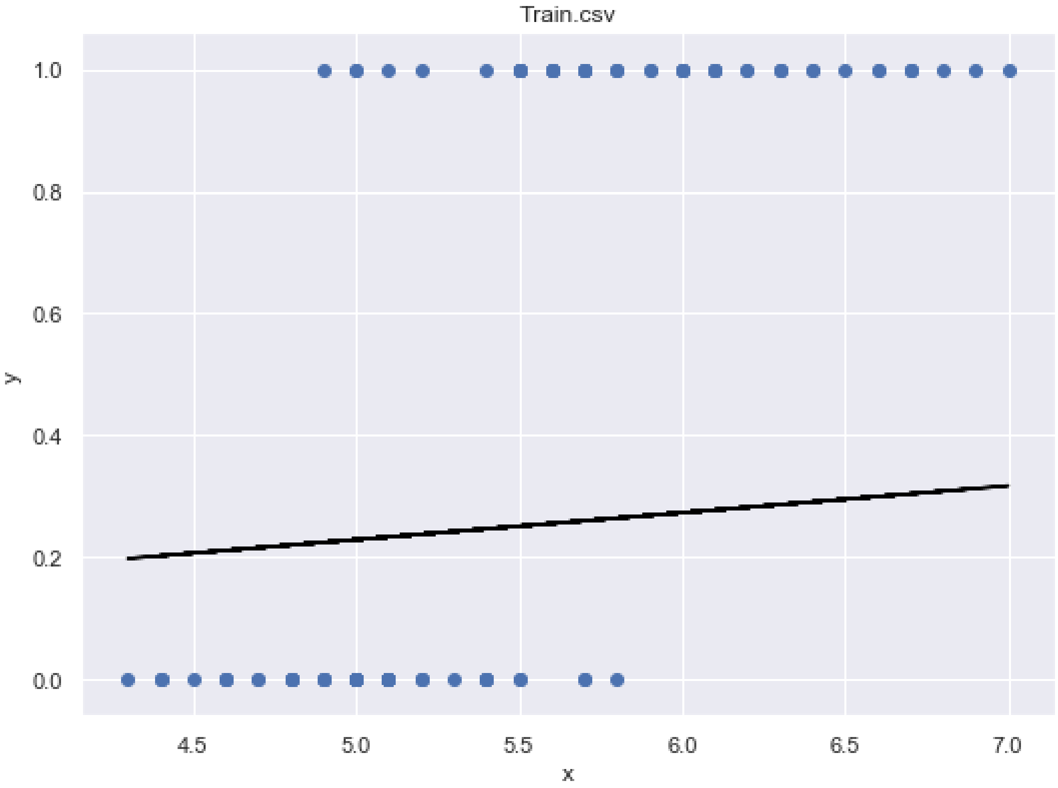
```
from sklearn import datasets
import numpy as np
from matplotlib import pyplot as plt
iris = datasets.load_iris()
X = iris.data[:100, :4]
y = iris.target[:100]
x_test=iris.data[101:,:1]
print("SHAPE OF X:",X.shape,y.shape,x_test.shape)
x=[]
xtest=[]
for i in X:
    x.append(i[0])
for i in x_test:
    xtest.append(i[0])
x=np.array(x)
y=np.array(y)
print(x)
print(y)
```

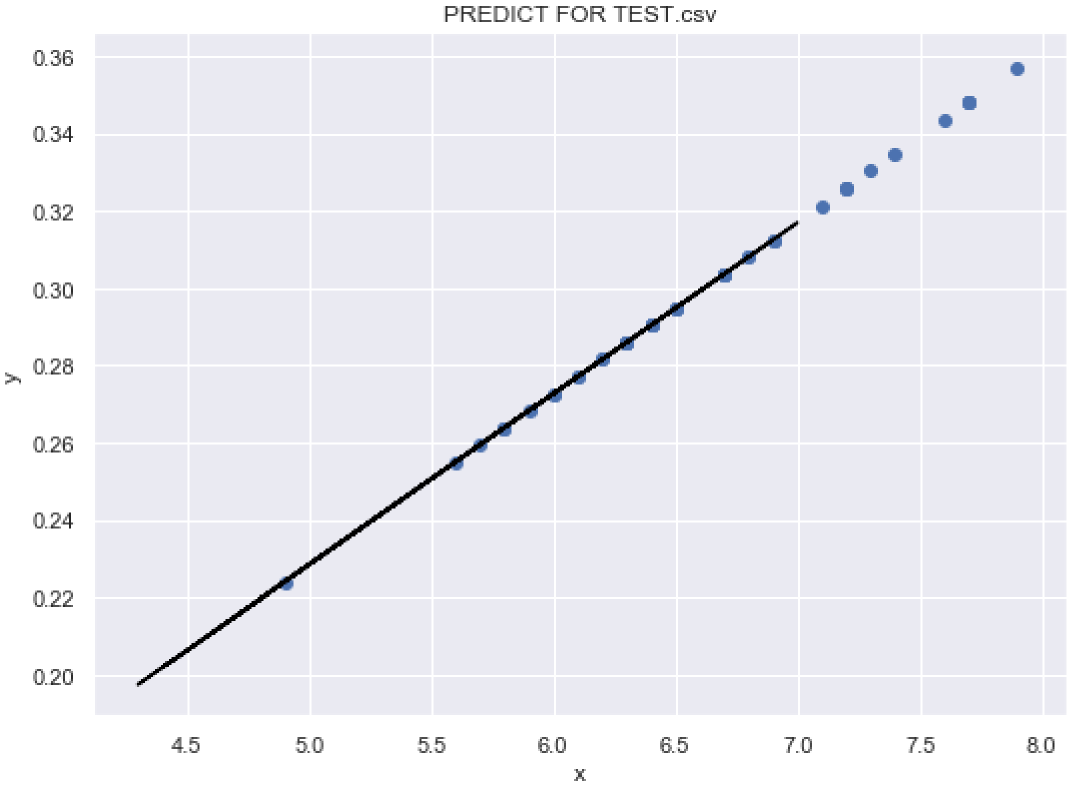
```
SHAPE OF X: (100, 4) (100,) (49, 1)
[5.1 4.9 4.7 4.6 5.  5.4 4.6 5.  4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.  5.  5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.
 5.5 4.9 4.4 5.1 5.  4.5 4.4 5.  5.1 4.8 5.1 4.6 5.3 5.  7.  6.4 6.9 5.5
 6.5 5.7 6.3 4.9 6.6 5.2 5.  5.9 6.  6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
 6.3 6.1 6.4 6.6 6.8 6.7 6.  5.7 5.5 5.5 5.8 6.  5.4 6.  6.7 6.3 5.6 5.5
 5.5 6.1 5.8 5.  5.6 5.7 5.7 6.2 5.1 5.7]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [62]:

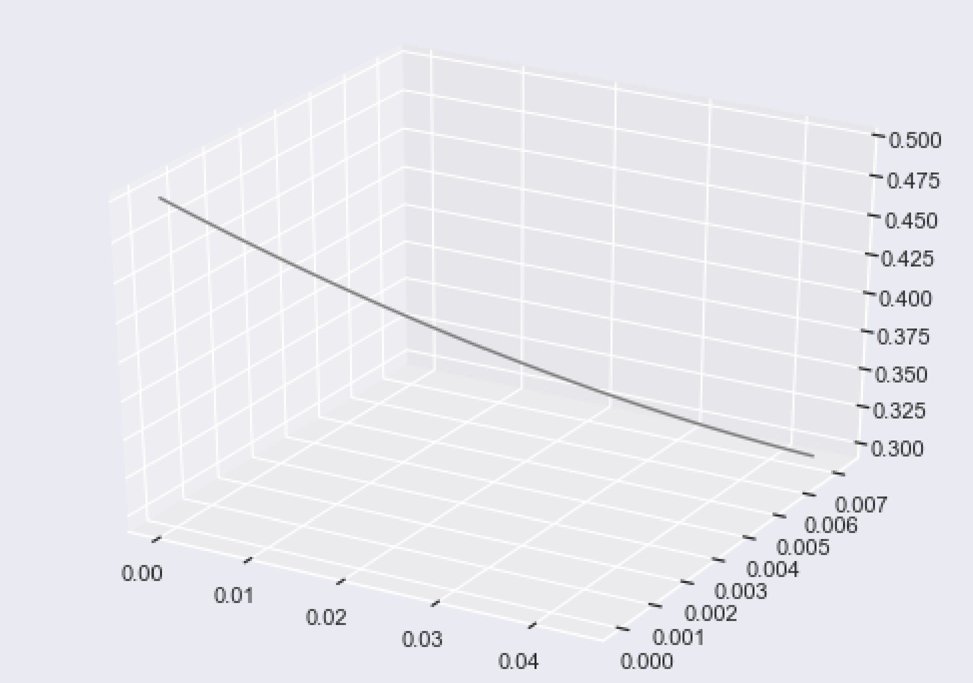
```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
from mpl_toolkits import mplot3d

def gradient_descent(x,y,x_find):
    m=b=0
    iteration=100
    n=len(x)
    print("length={}".format(n))
    alpha=0.0001
    error=[];m_=[];b_=[]

    for i in range(iteration):
        m_.append(m)
        b_.append(b)
        y_pred=m*x+b
        cost=(1/n)*sum([val**2 for val in (y-y_pred)])
        error.append(cost)
        md=(2/n)*sum(x*(y_pred-y))
        bd=(2/n)*sum(y_pred-y)
        b=b-alpha*bd
        m=m-alpha*md
    print("m={} and b={}".format(m,b))
    plt.scatter(x,y)
    plt.plot(x,m*x+b,color="black")
    plt.title("Train.csv")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()
    plt.plot(x,m*x+b,color="black")
    plt.scatter(x_find,m*x_find+b)
    plt.title("PREDICT FOR TEST.csv")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()
    print("3D COST FUNCTION")
    ax = plt.axes(projection='3d')
    ax.plot3D(m_, b_,error, 'gray')
    plt.show()



x_find=np.array(xtest)
print(x)
print(y)
gradient_descent(x,y,x_find)
```

```
[5.1 4.9 4.7 4.6 5.  5.4 4.6 5.  4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.  5.  5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.
 5.5 4.9 4.4 5.1 5.  4.5 4.4 5.  5.1 4.8 5.1 4.6 5.3 5.  7.  6.4 6.9 5.5
 6.5 5.7 6.3 4.9 6.6 5.2 5.  5.9 6.  6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
 6.3 6.1 6.4 6.6 6.8 6.7 6.  5.7 5.5 5.5 5.8 6.  5.4 6.  6.7 6.3 5.6 5.5
 5.5 6.1 5.8 5.  5.6 5.7 5.7 6.2 5.1 5.7]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
length=100
m=0.0442400506866422 and b=0.007272401695566481
```


Train.csv
y
1.0
0.8
0.6
0.4
0.2
0.0
x
4.5
5.0
5.5
6.0
6.5
7.0


PREDICT FOR TEST.csv
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20
y
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
x

3D COST FUNCTION

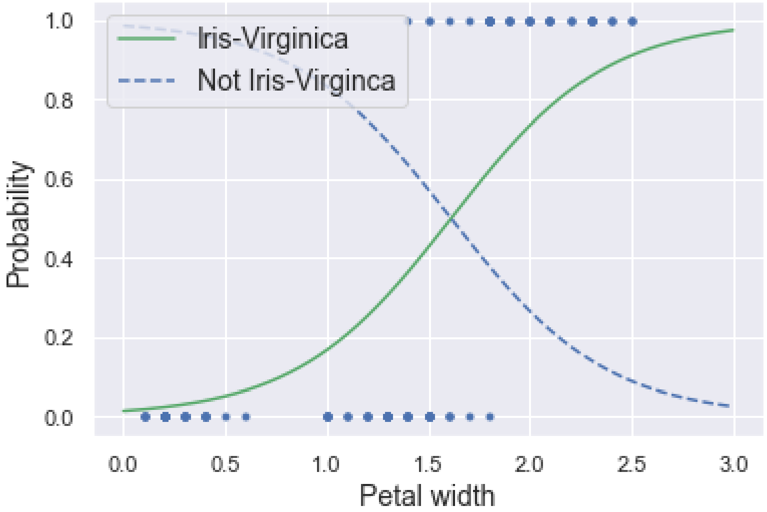

0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0.00
0.01
0.02
0.03
0.04

Logistic regression on IRIS Dataset

In [67]:

```
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
import numpy as np

%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

iris = datasets.load_iris()

print(list(iris.keys()))

#print(iris)

X = iris["data"][:,3:]  # petal width
y = (iris["target"]==2).astype(np.int)

log_reg = LogisticRegression(penalty="l2")
log_reg.fit(X,y)

X_new = np.linspace(0,3,1000).reshape(-1,1)
y_proba = log_reg.predict_proba(X_new)

plt.plot(X,y,"b.")
plt.plot(X_new,y_proba[:,1],"g-",label="Iris-Virginica")
plt.plot(X_new,y_proba[:,0],"b--",label="Not Iris-Virginca")
plt.xlabel("Petal width", fontsize=14)
plt.ylabel("Probability", fontsize=14)
plt.legend(loc="upper left", fontsize=14)
plt.show()

log_reg.predict([[1.7],[1.5]])
```

```
['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']
```


Iris-Virginica
Not Iris-Virginca
Probability
Petal width
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

```
Out[67]:

array([1, 0])
```

K Mean Clustering

k-means clustering with k = 3

In [66]:

```
import numpy
from sklearn import datasets,metrics
from sklearn.cluster import KMeans
x,y=datasets.load_iris(return_X_y=True)
x_train=x[range(0,150,2),:]
x_test=x[range(1,150,2),:]
k_model=KMeans(n_clusters=3)
k_model.fit(x_train)
print(k_model.labels_)
pred=k_model.predict(x_test)
print("\nPREDICTION FOR X_TEST :",pred)
print("\nCLUSTER CENTERS :\n",k_model.cluster_centers_)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 2 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 0 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 0 2 0 2 2
 2]

PREDICTION FOR X_TEST : [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0
 0 2 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 2 2 0 0 0 2 0 2 2 0 2 2 2 2 2 2 2
 0]

CLUSTER CENTERS :
 [[5.92142857 2.76071429 4.35714286 1.41071429]
 [5.024      3.48       1.456      0.228     ]
 [6.66363636 2.97727273 5.67272727 2.1       ]]
```

Conclusion

After implementing this practical I have understood ML techniques for the Big data and what are the problems arises and how to overcome the problems. I have learnt difference between linear regression and logistic regression.

```
In [ ]:
```