Machine learning Important Algorithm

Explain Concept of **Regression**

With **Linear Regression**

What Is Regression?

Regression searches for relationships among **variables**. For example, you can observe several employees of some company and try to understand how their salaries depend on their **features**, such as experience, education level, role, city of employment, and so on.

This is a regression problem where data related to each employee represents one **observation**. The presumption is that the experience, education, role, and city are the independent features, while the salary depends on them.

The dependent features are called the **dependent variables**, **outputs**, or **responses**. The independent features are called the **independent variables**, **inputs**, **regressors**, or **predictors**

When Do You Need Regression?

Regression is also useful when you want to **forecast** a response using a new set of predictors. For example, you could try to predict electricity consumption of a household for the next hour given the outdoor temperature, time of day, and number of residents in that household.

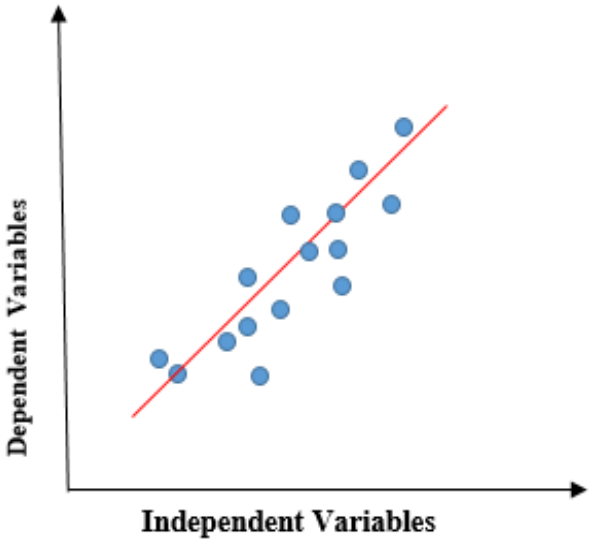
Types of Regression models

1. Linear Regression
2. Polynomial Regression
3. Logistics Regression

1 Linear Regression

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression. If there is a single input variable (x), such linear regression is called simple linear regression. And if there is more than one input variable, such linear regression is called multiple linear regression. The linear regression model gives a sloped straight line describing the relationship within the variables.

1 Linear Regression



The above graph presents the linear relationship between the dependent variable and independent variables. When the value of x (independent variable) increases, the value of y (dependent variable) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best.

1 Linear Regression

$$y = mx+b \quad \Longrightarrow \quad y = a_0+a_1x$$

y= Dependent Variable.

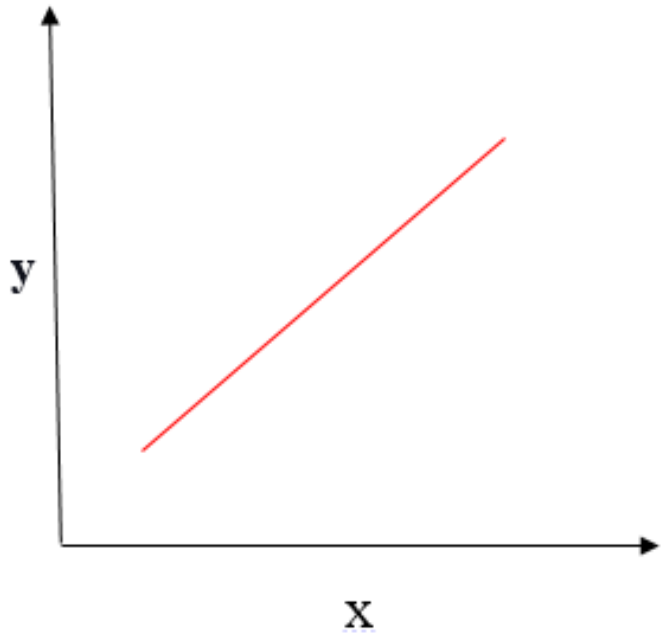
x= Independent Variable.

a0= intercept of the line.

a1 = Linear regression coefficient.

1 Positive Linear Relationship

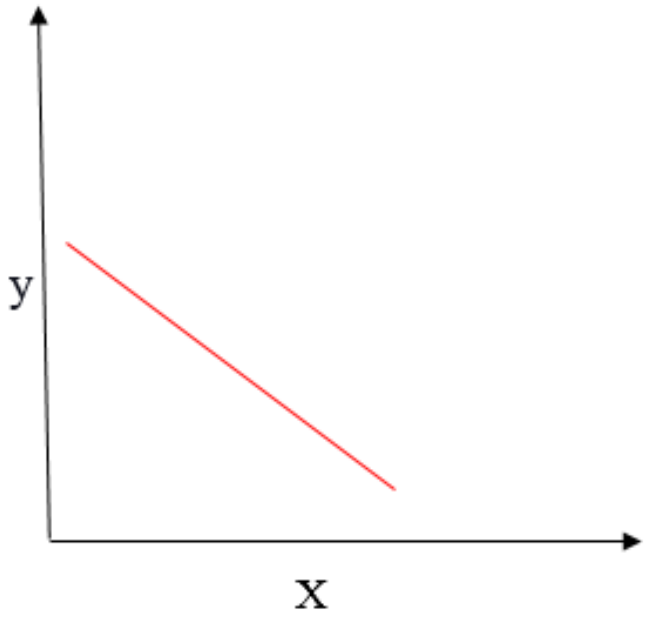
If the dependent variable expands on the Y-axis and the independent variable progress on X-axis, then such a relationship is termed a Positive linear relationship.



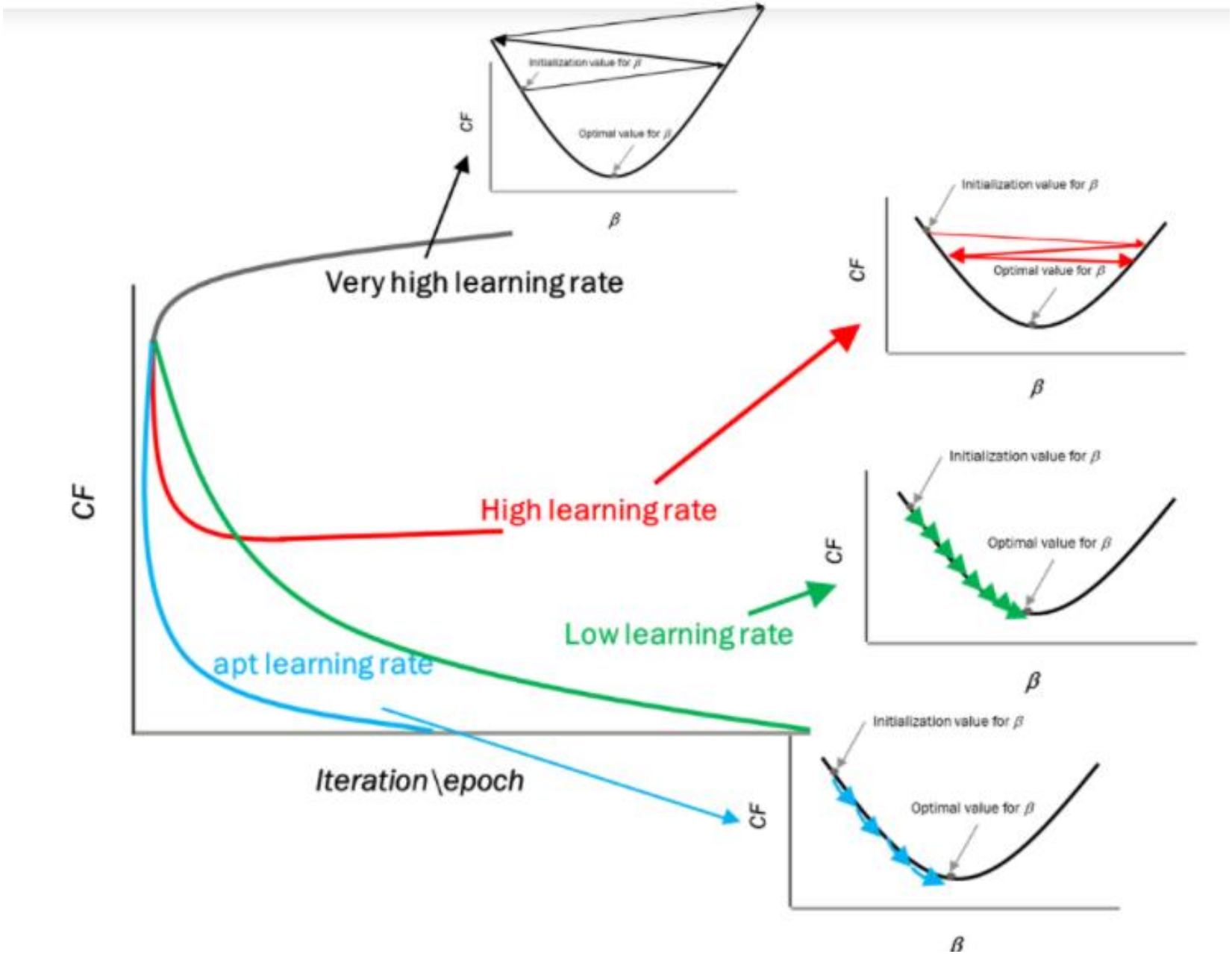
$$y = a_0 + a_1 x$$

1 Negative Linear Relationship

If the dependent variable decreases on the Y-axis and the independent variable increases on the X-axis, such a relationship is called a negative linear relationship.



Impact of different values for learning rate

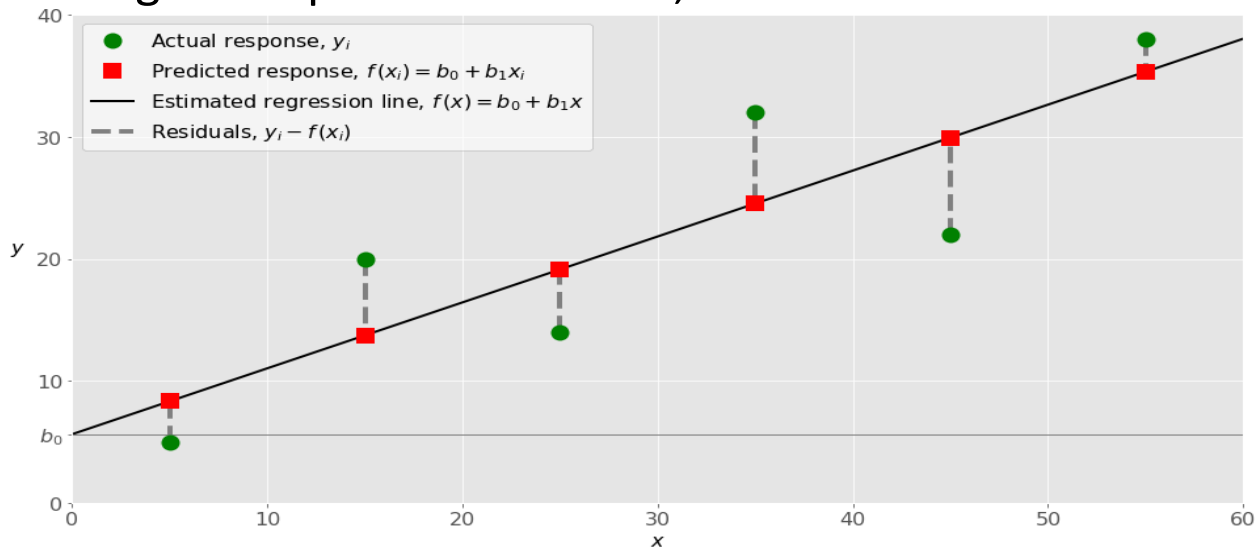


Linear Regression

Linear regression is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results.

Simple Linear Regression

Simple or single-variate linear regression is the simplest case of linear regression, as it has a single independent variable, $\mathbf{x} = x$.



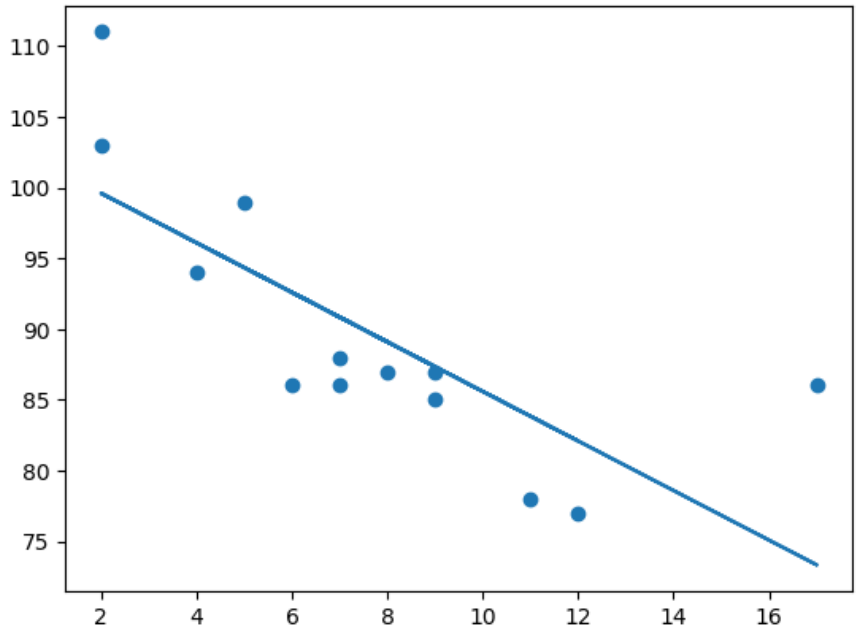
Python Code For Linear Regression

```
from sklearn import linear_model

features = [[2],[1],[5],[10]]

labels = [27, 11, 75, 155]

clf = linear_model.LinearRegression()

clf=clf.fit(features,labels)

#predicted = clf.predict([[5]])

predicted = clf.predict([[5], [5], [2]])

print(predicted)
```

Output :[75. 75. 27.]

Python Code For Linear Regression

```
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y) # use of Linerreg function

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```



Python Code For Logistic Regression

```
# import the necessary libraries
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True)
# split the train and test dataset
X_train, X_test,\
            y_train, y_test = train_test_split(X, y,

            test_size=0.20,

            random_state=23)
# LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)
# Prediction
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Logistic Regression model accuracy (in %):", acc*100)
```

End of Topic