Data Science & AI

Machine Learning

Practical Implementation

Lecture No.- 01

By- Krish Naik Sir

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
df=pd.read_csv('Algerian_forest_fires_dataset_UPDATE (8).csv')
```

```
df.head()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	1	6	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire
4	5	6	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire

```
df.columns
```

```
Index(['day', 'month', 'year', 'Temperature', ' RH', ' Ws', 'Rain ', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes  '],
      dtype='object')
```

```
##drop month,day and yyear
df.drop(['day','month','year'],axis=1,inplace=True)
```

```
df.head()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire
1	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire
3	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire
4	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire

```
df.head()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire
1	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire
3	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire
4	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire

```
df['Classes  '].value_counts()
```

```
1    138
0    109
Name: Classes  , dtype: int64
```

```
df['Classes']=df['Classes  ']
df.drop(['Classes  '],axis=1,inplace=True)
```

```
## Encoding
df['Classes  ']=np.where(df['Classes  '].str.contains("not fire"),0,1)
```

```
df['Classes'].value_counts()
```

```
1    138
0    109
Name: Classes, dtype: int64
```

```
df.tail()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
242	30	65	14	0	85.4	16	44.5	4.5	16.9	6.5	1
243	28	87	15	4.4	41.1	6.5	8	0.1	6.2	0	0
244	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	0
245	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	0
246	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	0

```
df['Classes'].value_counts()
```

```
1    137
0    106
Name: Classes, dtype: int64
```

```
## Independent And dependent features
X=df.drop('FWI',axis=1) #independent
y=df['FWI'] #dependent
```

```
X.head()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes
0	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0
1	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0
3	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0
4	27	77	16	0	64.8	3	14.2	1.2	3.9	0

```
y
```

```
0      0.5
1      0.4
2      0.1
3        0
4      0.5
      ... 
242    6.5
243      0
244    0.2
245    0.7
246    0.5
Name: FWI, Length: 247, dtype: object
```

```
#Train Test Split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

```
X_train.shape,X_test.shape
```

```
((185, 10), (62, 10))
```

```
X_train.head()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes
101	33	73	12	1.8	59.9	2.2	8.9	0.7	2.7	0
197	39	21	17	0.4	93	18.4	41.5	15.5	18.4	1
126	30	73	13	4	55.7	2.7	7.8	0.6	2.9	0
69	35	59	17	0	87.4	14.8	57	6.9	17.9	1
200	35	46	13	0.3	83.9	16.9	54.2	3.5	19	1

```
X_train[X_train['Temperature']== 'Temperature']
```

Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes

```
X_train.drop(index=124,inplace=True)
```

```
X_train.corr()
```

```
<ipython-input-60-1d31ae5364df>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  X_train.corr()
```

index	Classes
Classes	

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

```
X_train[X_train.Temperature !='Temperature'].corr()
```

```
<ipython-input-54-d6f31b53d14c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  X_train[X_train.Temperature != 'Temperature'].corr()
```

	Classes
Classes	1.0

```
X_train.columns
```

```
Index(['Temperature', ' RH', ' Ws', 'Rain ', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI',
       'Classes'],
      dtype='object')
```

```
X_train['Temperature']=X_train['Temperature'].dropna().astype(int)
```

```
X_train[' RH']=X_train[' RH'].dropna().astype(int)
```

```
X_train[' Ws']=X_train[' Ws'].dropna().astype(int)
X_train['Rain ']=X_train['Rain '].dropna().astype(float)
X_train['FFMC']=X_train['FFMC'].dropna().astype(float)
X_train['DMC']=X_train['DMC'].dropna().astype(float)
#X_train['DC']=X_train['DC'].dropna().astype(float)
X_train['ISI']=X_train['ISI'].dropna().astype(float)
X_train['BUI']=X_train['BUI'].dropna().astype(float)
```

```
X_train['DC']=X_train['DC'].dropna().replace('14.6 9','14.69').astype(float)
```

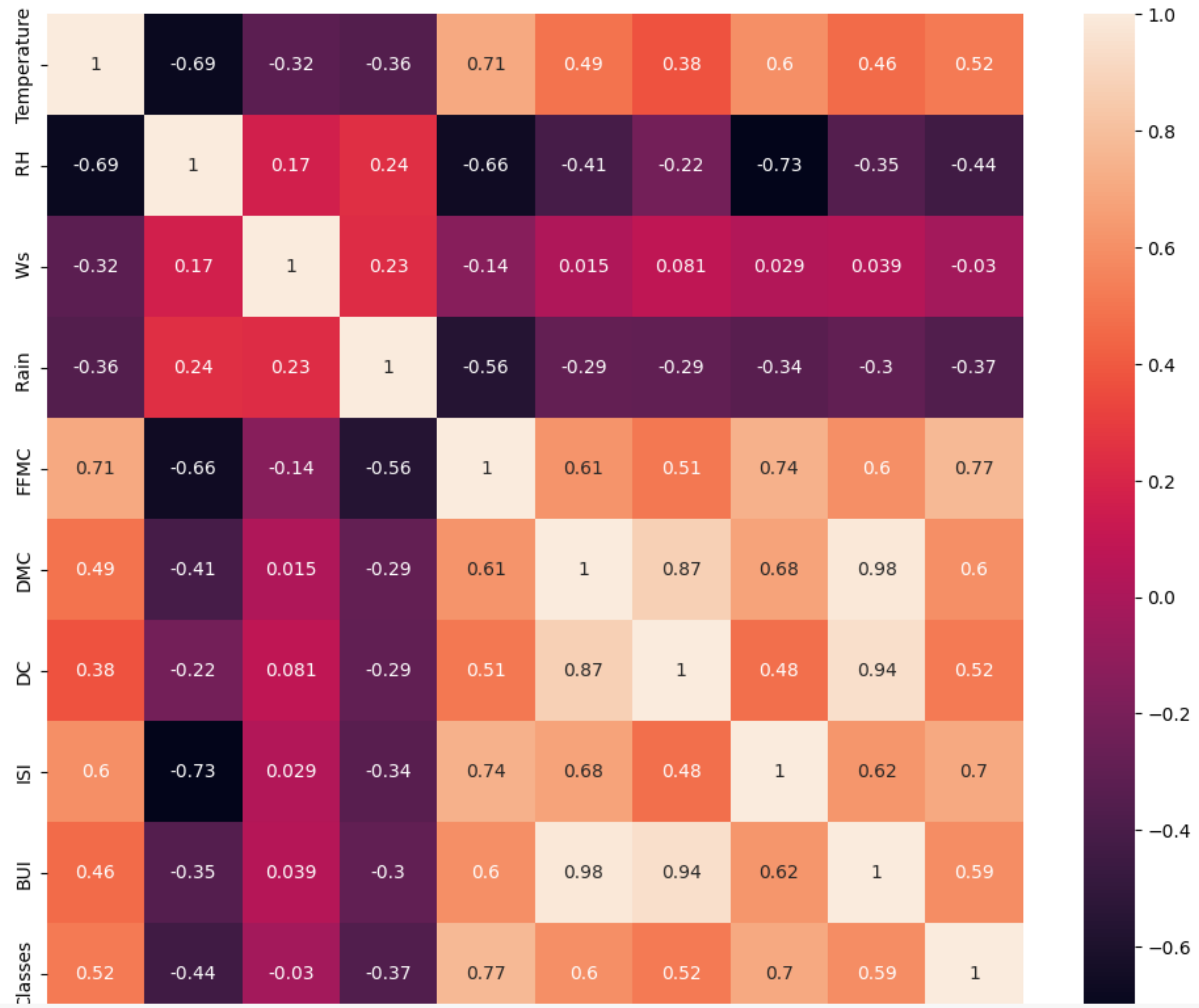
```
X_train.corr()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes
Temperature	1.000000	-0.689393	-0.321891	-0.359438	0.707745	0.490281	0.376328	0.598660	0.463008	0.515195
RH	-0.689393	1.000000	0.166559	0.244101	-0.660022	-0.410668	-0.219077	-0.732962	-0.352303	-0.438307
Ws	-0.321891	0.166559	1.000000	0.229595	-0.141418	0.015022	0.081155	0.029341	0.039326	-0.030138
Rain	-0.359438	0.244101	0.229595	1.000000	-0.557421	-0.286336	-0.294696	-0.337800	-0.295782	-0.365927
FFMC	0.707745	-0.660022	-0.141418	-0.557421	1.000000	0.614965	0.510088	0.740773	0.597772	0.773751
DMC	0.490281	-0.410668	0.015022	-0.286336	0.614965	1.000000	0.871724	0.676476	0.983552	0.599769
DC	0.376328	-0.219077	0.081155	-0.294696	0.510088	0.871724	1.000000	0.475461	0.943763	0.517169
ISI	0.598660	-0.732962	0.029341	-0.337800	0.740773	0.676476	0.475461	1.000000	0.623201	0.703945
BUI	0.463008	-0.352303	0.039326	-0.295782	0.597772	0.983552	0.943763	0.623201	1.000000	0.591169
Classes	0.515195	-0.438307	-0.030138	-0.365927	0.773751	0.599769	0.517169	0.703945	0.591169	1.000000

Feature Selection

```
## Check for multicollinearity
plt.figure(figsize=(12,10))
corr=X_train.corr()
sns.heatmap(corr,annot=True)
```

<Axes: >




```
X_train.corr()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes
Temperature	1.000000	-0.689393	-0.321891	-0.359438	0.707745	0.490281	0.376328	0.598660	0.463008	0.515195
RH	-0.689393	1.000000	0.166559	0.244101	-0.660022	-0.410668	-0.219077	-0.732962	-0.352303	-0.438307
Ws	-0.321891	0.166559	1.000000	0.229595	-0.141418	0.015022	0.081155	0.029341	0.039326	-0.030138
Rain	-0.359438	0.244101	0.229595	1.000000	-0.557421	-0.286336	-0.294696	-0.337800	-0.295782	-0.365927
FFMC	0.707745	-0.660022	-0.141418	-0.557421	1.000000	0.614965	0.510088	0.740773	0.597772	0.773751
DMC	0.490281	-0.410668	0.015022	-0.286336	0.614965	1.000000	0.871724	0.676476	0.983552	0.599769
DC	0.376328	-0.219077	0.081155	-0.294696	0.510088	0.871724	1.000000	0.475461	0.943763	0.517169
ISI	0.598660	-0.732962	0.029341	-0.337800	0.740773	0.676476	0.475461	1.000000	0.623201	0.703945
BUI	0.463008	-0.352303	0.039326	-0.295782	0.597772	0.983552	0.943763	0.623201	1.000000	0.591169
Classes	0.515195	-0.438307	-0.030138	-0.365927	0.773751	0.599769	0.517169	0.703945	0.591169	1.000000

```
def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

```
## threshold--Domain expertise
corr_features=correlation(X_train,0.85)
```

```
corr_features
```

```
{'BUI', 'DC'}
```

```
## drop features when correlation is more than 0.85
X_train.drop(corr_features,axis=1,inplace=True)
X_test.drop(corr_features,axis=1,inplace=True)
X_train.shape,X_test.shape
```

```
((184, 8), (62, 8))
```

Feature Scaling Or Standardization

```
X_train.dropna(axis=0).isnull().sum()
```

```
Temperature    0
 RH            0
 Ws            0
Rain           0
FFMC           0
DMC            0
ISI            0
Classes        0
dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train.dropna(axis=0))
X_test_scaled=scaler.transform(X_test)
```

```
X_train_scaled
```

```
array([[ 0.18091033,  0.71998514, -1.34871966, ..., -0.9988569 ,
        -0.95996409, -1.14183951],
       [ 1.78704105, -2.7887375 ,  0.57500274, ...,  0.27897956,
         2.42143943,  0.87577982],
       [-0.62215503,  0.71998514, -0.96397518, ..., -0.95941751,
        -0.98281141, -1.14183951],
       ...,
       [-1.9605973 ,  0.92241145,  0.57500274, ..., -1.06984782,
        -1.07420069, -1.14183951],
       [ 1.78704105,  0.11270622, -2.5029531 , ..., -0.24950836,
        -0.8685748 , -1.14183951],
       [-0.62215503,  0.98988688,  2.11398066, ..., -1.02252054,
        -0.8685748 , -1.14183951]])
```

Box Plots To understand Effect Of Standard Scaler

```
plt.subplots(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.boxplot(data=X_train)
plt.title('X_train Before Scaling')
plt.subplot(1, 2, 2)
sns.boxplot(data=X_train_scaled)
plt.title('X_train After Scaling')
```

```
<ipython-input-96-41fb1d7ced73>:2: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will b
  plt.subplot(1, 2, 1)
Text(0.5, 1.0, 'X_train After Scaling')
```


```
pd.DataFrame(X_train_scaled).isnull().sum()
```

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
dtype: int64
```

```
pd.DataFrame(X_test_scaled).isnull().sum()
```

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
dtype: int64
```

```
y_train=y_train.replace('FWI','0').replace('fire   ','0').astype(float)
```

```
y_train.dropna(inplace=True)
```

```
y_train.shape
```

```
(183,)
```

Linear Regression Model

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
linreg=LinearRegression()
linreg.fit(X_train_scaled,y_train[1:])
y_pred=linreg.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 1.1001680700952507
R2 Score 0.9375294317383766
```

Lasso Regression

```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
lasso=Lasso()
lasso.fit(X_train_scaled,y_train[1:])
y_pred=lasso.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 1.6622251402428814
R2 Score 0.891008091956091
```

Ridge Regression model

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error
```

```
from sklearn.metrics import r2_score
ridge=Ridge()
ridge.fit(X_train_scaled,y_train[1:])
y_pred=ridge.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 1.1010123032721502
R2 Score 0.9372669856655736
```

Elasticnet Regression

```
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
elastic=ElasticNet()
elastic.fit(X_train_scaled,y_train[1:])
y_pred=elastic.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 1.9749711385449351
R2 Score 0.8436374971301746
```

```
import pickle
pickle.dump(scaler,open('scaler.pkl','wb'))
pickle.dump(ridge,open('ridge.pkl','wb'))
```

Logistic Regression Implementation

```
from sklearn.datasets import load_iris
```

```
dataset=load_iris()
```

```
print(dataset.DESCR)
```

```
    **Data Set Characteristics:**

        :Number of Instances: 150 (50 in each of three classes)
        :Number of Attributes: 4 numeric, predictive attributes and the class
        :Attribute Information:
            - sepal length in cm
            - sepal width in cm
            - petal length in cm
            - petal width in cm
            - class:
                    - Iris-Setosa
                    - Iris-Versicolour
                    - Iris-Virginica

        :Summary Statistics:

        ============== ==== ==== ======= ===== ====================
                        Min  Max   Mean    SD   Class Correlation
        ============== ==== ==== ======= ===== ====================
        sepal length:   4.3  7.9   5.84   0.83    0.7826
        sepal width:    2.0  4.4   3.05   0.43   -0.4194
        petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
        petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
        ============== ==== ==== ======= ===== ====================

        :Missing Attribute Values: None
        :Class Distribution: 33.3% for each of 3 classes.
        :Creator: R.A. Fisher
        :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
        :Date: July, 1988

    The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
    from Fisher's paper. Note that it's the same as in R, but not as in the UCI
    Machine Learning Repository, which has two wrong data points.

    This is perhaps the best known database to be found in the
    pattern recognition literature.  Fisher's paper is a classic in the field and
    is referenced frequently to this day.  (See Duda & Hart, for example.)  The
    data set contains 3 classes of 50 instances each, where each class refers to a
    type of iris plant.  One class is linearly separable from the other 2; the
    latter are NOT linearly separable from each other.

    .. topic:: References

       - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
         Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
         Mathematical Statistics" (John Wiley, NY, 1950).
       - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
         (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
       - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
         Structure and Classification Rule for Recognition in Partially Exposed
         Environments".  IEEE Transactions on Pattern Analysis and Machine
         Intelligence, Vol. PAMI-2, No. 1, 67-71.
       - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
         on Information Theory, May 1972, 431-433.
       - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
         conceptual clustering system finds 3 classes in the data.
       - Many, many more ...
```

```
dataset.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
import pandas as pd
import numpy as np
```

```
df=pd.DataFrame(dataset.data,columns=dataset.feature_names)
```

```
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2

```
df['target']=dataset.target
```

```
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
dataset.target
```

```
array([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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
## Binary Classification
df_copy=df[df['target']!=2]
```

```
df_copy.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Independent and dependent features

```
X=df_copy.iloc[:,:-1]
y=df_copy.iloc[:,-1]
```

```
from sklearn.linear_model import LogisticRegression
```

```
#train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=20, random_state=42)
```

```
classifier=LogisticRegression()
```

```
classifier.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
classifier.predict_proba(X_test)
```

```
array([[0.00118085, 0.99881915],
       [0.01580857, 0.98419143],
       [0.00303433, 0.99696567],
       [0.96964813, 0.03035187],
```

```
       [0.94251523, 0.05748477],
       [0.97160984, 0.02839016],
       [0.99355615, 0.00644385],
       [0.03169836, 0.96830164],
       [0.97459743, 0.02540257],
       [0.97892756, 0.02107244],
       [0.95512297, 0.04487703],
       [0.9607199 , 0.0392801 ],
       [0.00429472, 0.99570528],
       [0.9858324 , 0.0141676 ],
       [0.00924893, 0.99075107],
       [0.98144334, 0.01855666],
       [0.00208036, 0.99791964],
       [0.00125422, 0.99874578],
       [0.97463766, 0.02536234],
       [0.96123726, 0.03876274]])
```

```
## Prediction
y_pred=classifier.predict(X_test)
```

```
y_pred
```

```
array([1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0])
```

```
y_test
```

```
83    1
53    1
70    1
45    0
44    0
39    0
22    0
80    1
10    0
0     0
18    0
30    0
73    1
33    0
90    1
4     0
76    1
77    1
12    0
31    0
Name: target, dtype: int64
```

Confusion matrix,accuracy score,classification report

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

```
print(confusion_matrix(y_pred,y_test))
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

```
[[12  0]
 [ 0  8]]
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      1.00      1.00         8

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Hyperparameter Tuning

```
## Gridsearchcv
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
```

```
parameters={'penalty':('l1','l2','elasticnet',None),'C':[1,10,20]}
```

```
clf=GridSearchCV(classifier,param_grid=parameters,cv=5)
```

```
## Splitting of Train data to validation data
clf.fit(X_train,y_train)
```

```
GridSearchCV
estimator: LogisticRegression
LogisticRegression
```

```
clf.best_params_
```

```
{'C': 1, 'penalty': 'l2'}
```

```
classifier=LogisticRegression(C=1,penalty='l2')
```

```
classifier.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(C=1)
```

```
## Prediction
y_pred=classifier.predict(X_test)
print(confusion_matrix(y_pred,y_test))
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

```
[[12  0]
 [ 0  8]]
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      1.00      1.00         8

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

```
## Reandomized Searchcv
from sklearn.model_selection import RandomizedSearchCV
```

```
random_clf=RandomizedSearchCV(LogisticRegression(),param_distributions=parameters,cv=5)
```

```
random_clf.fit(X_train,y_train)
```

```
RandomizedSearchCV
estimator: LogisticRegression
LogisticRegression
```

```
random_clf.best_params_
```

```
{'penalty': None, 'C': 10}
```

```
## Logistic Regression Create And check Accuracy
```

THANK - YOU