Iris Dataset Classification using K nearest neighbors and Logistic Regression

```
import sklearn
```

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

```
from sklearn import datasets
```

```
df = datasets.load_iris()
print(type(df))
X = df.data
y = df.target
print(X.shape)
print(y.shape)
data1 = pd.DataFrame(data= np.c_[df['data'], df['target']],
                     columns= df['feature_names'] + ['target'])
print(data1.head())
```

```
<class 'sklearn.utils.Bunch'>
(150, 4)
(150,)
   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
4                5.0               3.6                1.4               0.2

   target
0     0.0
1     0.0
2     0.0
3     0.0
4     0.0
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

```
print(x_train.shape,x_test.shape)
print(y_train.shape,y_test.shape)
```

```
(120, 4) (30, 4)
(120,) (30,)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
knn = KNeighborsClassifier(n_neighbors=3)
lgr = LogisticRegression(random_state=0).fit(x_train,y_train)
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_predictionsKNN = knn.predict(x_test)
y_predictionsLGR = lgr.predict(x_test)
```

```
print("Accuracy in KNN algo : ",metrics.accuracy_score(y_test,y_predict
print("Accuracy in LGR algo : ",metrics.accuracy_score(y_test,y_predict
```

```
Accuracy in KNN algo :  1.0
Accuracy in LGR algo :  0.9666666666666667
```

```
print(df.target_names)
```

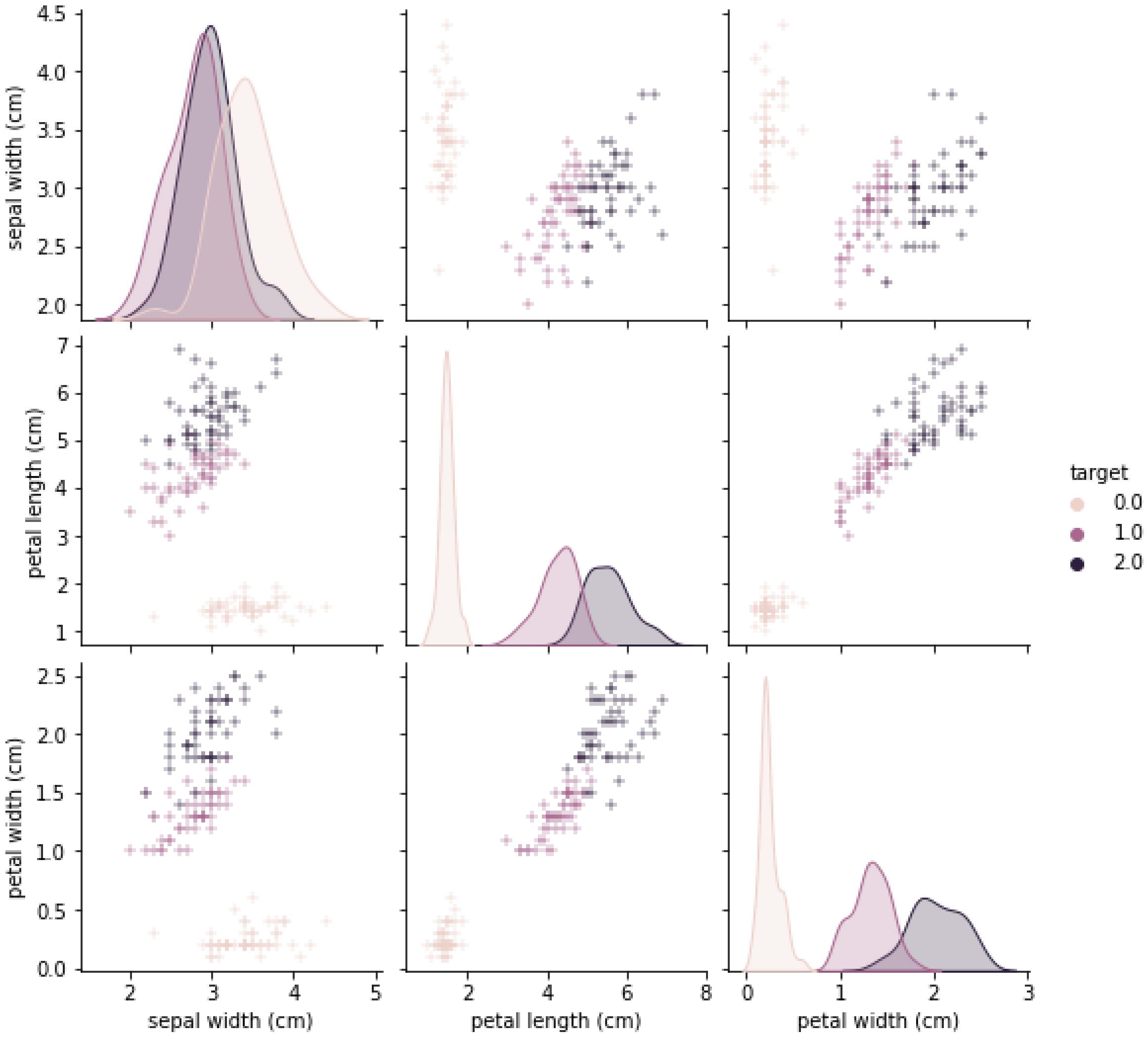
```
['setosa' 'versicolor' 'virginica']
```

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    float64
dtypes: float64(5)
memory usage: 6.0 KB
```

```
data1.describe()
```

```
import seaborn as sns
tmp = data1.drop('sepal length (cm)', axis=1)
g = sns.pairplot(tmp, hue='target', markers='+')
plt.show()
```



House Price detection using Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score


raw_house = pd.read_csv('/content/data.csv')
#raw_house
raw_house = raw_house.dropna()
raw_house
```

```
raw_house_dup = raw_house
raw_house_dup
```

```
# Checking outliers from the datasets
# Box plot
#
sns.boxplot(raw_house.price)
```

```
#
# Distribution plot
#
sns.distplot(raw_house.price)
```

```
house_data = raw_house

corr_matrix = house_data.corr()
print(corr_matrix["price"].sort_values(ascending=False))
house_data = house_data[['price','bathrooms', 'sqft_living','sqft_above
print(house_data)
#sns.heatmap(house_data.corr(), annot=False)
sns.pairplot(house_data)
```

```
house_data.price.describe()
```

```
count    4.600000e+03
mean     5.519630e+05
std      5.638347e+05
min      0.000000e+00
25%      3.228750e+05
50%      4.609435e+05
75%      6.549625e+05
max      2.659000e+07
Name: price, dtype: float64
```

```
final_untouched_data = house_data[-20:]
```

```
training_data = house_data[:-20]
#training_data
```

```
feature_data = training_data.drop(['price'], axis=1)
target_data = training_data['price']
```

```
X_train, X_test, y_train, y_test = train_test_split(feature_data, target_
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((3206, 5), (1374, 5), (3206,), (1374,))
```

```
lr = LinearRegression()
lr.fit(X_train,y_train)
```

```
LinearRegression()
```

```
y_pred = lr.predict(X_test)
print(y_pred.shape)
```

```
(1374,)
```

```
print("Accuracy : ",lr.score(X_test,y_test))
```

```
Accuracy :  0.39593489264366966
```