Python-230901103 / 230901103 SKILL TEST -3.ipynb

Sundar-Sabari Add files via upload e15be7e · 1 minute ago

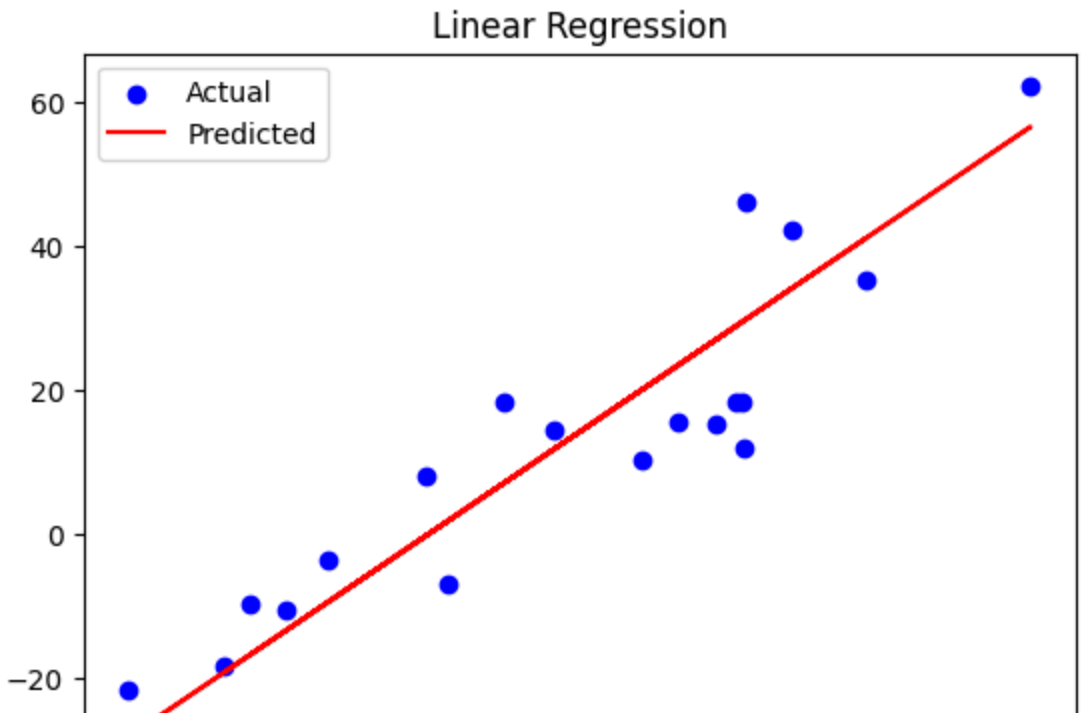
296 lines (296 loc) · 225 KB

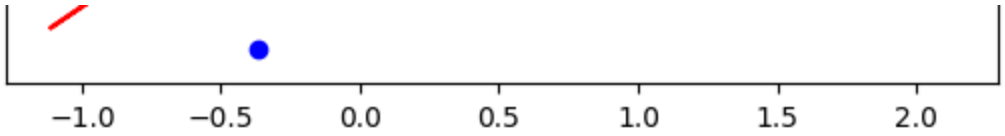
Preview Code Blame Raw

Q1: Create a Python Program to implement Linear Regression using any ML

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Sample dataset (You can replace this with your own CSV dataset)
from sklearn.datasets import make_regression
X, y = make_regression(n_samples=100, n_features=1, noise=10)

# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

# Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Plot
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Predicted')
plt.title("Linear Regression")
plt.legend()
plt.show()

# R² Score
print("Model Score (R²):", model.score(X_test, y_test))
```






```
Model Score (R²): 0.799541168893694
```

In [3]:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

# KNN Model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# Accuracy
print("KNN Accuracy:", accuracy_score(y_test, y_pred))
```

```
KNN Accuracy: 1.0
```

Q3: Write a Python program to detect outliers using Z-score method.

In [4]:

```
import numpy as np
import pandas as pd
from scipy.stats import zscore

# Sample data
data = {'value': [10, 12, 13, 12, 100, 14, 15, 16, 110, 17]}
df = pd.DataFrame(data)

# Calculate Z-scores
df['z_score'] = zscore(df['value'])

# Mark outliers
df['outlier'] = df['z_score'].apply(lambda x: abs(x) > 3)
print(df)
```

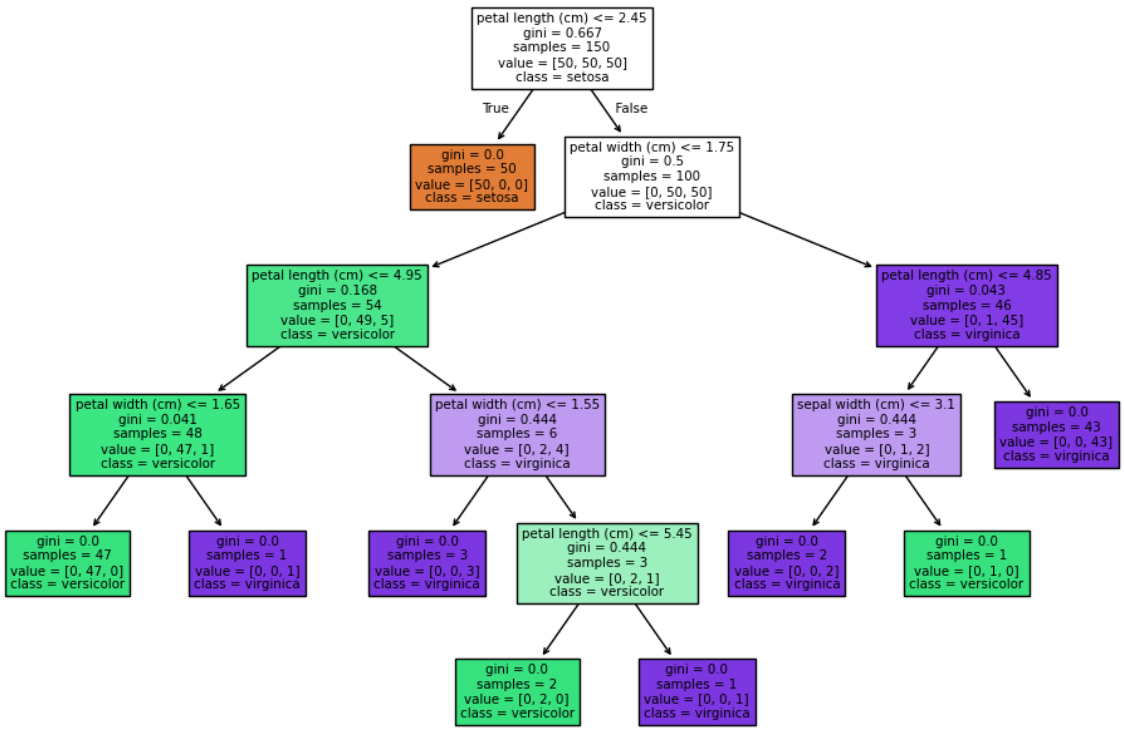
```
   value   z_score  outlier
0     10 -0.597218    False
1     12 -0.542678    False
2     13 -0.515407    False
```

```
3     12 -0.542678    False
4    100  1.857103    False
5     14 -0.488137    False
6     15 -0.460867    False
7     16 -0.433597    False
8    110  2.129806    False
9     17 -0.406327    False
```

Q4: Build a Decision Tree Classifier and display the tree.

In [5]:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Use Iris Dataset
iris = load_iris()
X = iris.data
y = iris.target

# Model
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Plot the tree
plt.figure(figsize=(12,8))
plot_tree(clf, feature_names=iris.feature_names, class_names=iris.target_names,
plt.show()
```



Q5: Write a program for K-means Clustering and plot the clusters.

In [6]:

```
from sklearn.datasets import make_blobs
```

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Sample clustered data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# KMeans Model
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

# Plotting
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200
plt.title("K-means Clustering")
plt.show()
```