JupyterLab | Jupyter Server | Home | Untitled | chatgpt login | Python Data

localhost:8889/notebooks/Untitled.ipynb?

jupyter Untitled Last Checkpoint: 3 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab | Python [conda env:base] * | Anaconda Toolbox

[1]:
```
# Step 1: Import necessary libraries
import numpy as np
import pandas as pd

# Step 2: Python Dictionary
# Creating a dictionary to store sample student data
student_dict = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [24, 22, 23],
    'Score': [85.5, 90.0, 88.5]
}
print("Dictionary:")
print(student_dict)

# Step 3: Convert Dictionary to List
# Creating list of tuples from dictionary
student_list = list(zip(student_dict['Name'], student_dict['Age'], student_dict['Score']))
print("\nList of Tuples:")
print(student_list)

# Step 4: Convert List to NumPy Array
student_array = np.array(student_list)
print("\nNumPy Array:")
print(student_array)

# Step 5: Convert Dictionary to Pandas DataFrame
student_df = pd.DataFrame(student_dict)
print("\nPandas DataFrame:")
print(student_df)

# Step 6: Display summary statistics (basic ML preprocessing)
print("\nDescriptive Statistics:")
print(student_df.describe())
```

```
Dictionary:
{'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [24, 22, 23], 'Score': [85.5, 90.0, 88.5]}

List of Tuples:
[('Alice', 24, 85.5), ('Bob', 22, 90.0), ('Charlie', 23, 88.5)]

NumPy Array:
[['Alice' '24' '85.5']
```

23°C Fog

Search

ENG IN

22:29 07-05-2025

JupyterLab | Jupyter Server | Home | Untitled | chatgpt login | Python Data

localhost:8889/notebooks/Untitled.ipynb?

jupyter Untitled Last Checkpoint: 4 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab | Python [conda env:base] * | Anaconda Toolbox

```
# Step 5: Convert Dictionary to Pandas DataFrame
student_df = pd.DataFrame(student_dict)
print("\nPandas DataFrame:")
print(student_df)

# Step 6: Display summary statistics (basic ML preprocessing)
print("\nDescriptive Statistics:")
print(student_df.describe())
```

```
Dictionary:
{'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [24, 22, 23], 'Score': [85.5, 90.0, 88.5]}

List of Tuples:
[('Alice', 24, 85.5), ('Bob', 22, 90.0), ('Charlie', 23, 88.5)]

NumPy Array:
[['Alice' '24' '85.5']
 ['Bob' '22' '90.0']
 ['Charlie' '23' '88.5']]

Pandas DataFrame:
      Name  Age  Score
0    Alice   24   85.5
1      Bob   22   90.0
2  Charlie   23   88.5

Descriptive Statistics:
             Age      Score
count   3.000000   3.000000
mean   23.000000  88.000000
std     1.000000   2.291288
min    22.000000  85.500000
25%    22.500000  87.000000
50%    23.000000  88.500000
75%    23.500000  89.250000
max    24.000000  90.000000
```

[ ]:

[ ]:

23°C Fog

Search

ENG IN

22:29 07-05-2025

Jupyte | Jupyte | Home | Untitle | ML pra | Untitle | Ur | chatgp | ML Pra | gdb cc | www.c

localhost:8889/notebooks/Untitled2.ipynb?

jupyter Untitled2 Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help

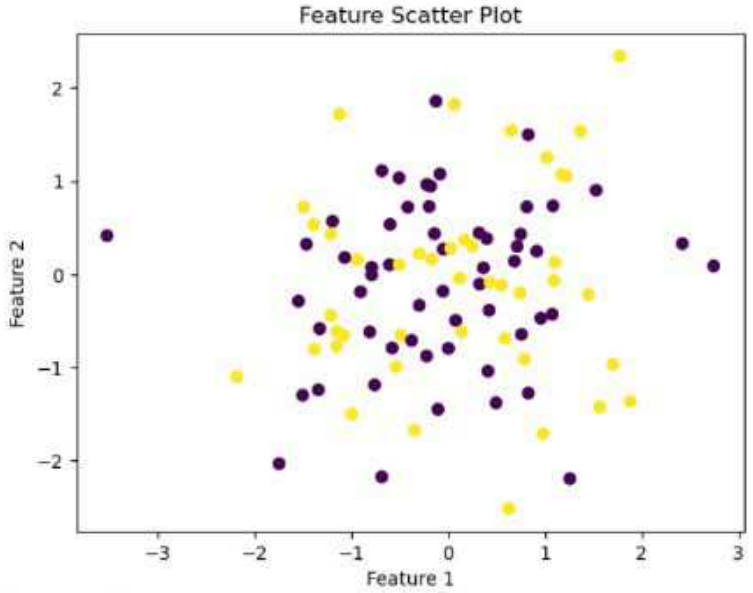
Trusted

Code

JupyterLab | Python [conda env:base] * | Anaconda Toolbox

```
# Sklearn: Train/test split, model training, and evaluation
X = df[['feature1', 'feature2']]
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
NumPy array: [0 1 2 3 4 5 6 7 8 9]
   feature1  feature2  label
0  0.119612 -0.043873      1
1 -0.604721  0.108932      0
2  1.881212 -1.364462      1
3 -1.324307 -0.584629      0
4  0.829369 -1.275497      0
```




```
Accuracy: 0.65
```

29°C Partly cloudy

Search

ENG IN

22:38 07-05-2025

Jupyte | Jupyte | Home | Untitle | ML pra | Untitle | Ur | chatgp | ML Pra | gdb cc | www.c

localhost:8889/notebooks/Untitled2.ipynb?

jupyter Untitled2 Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab | Python [conda env:base] * | Anaconda Toolbox

[1]:
```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# NumPy: Array creation and manipulation
arr = np.arange(10)
print("NumPy array:", arr)

# Pandas: Create DataFrame
df = pd.DataFrame({
    'feature1': np.random.randn(100),
    'feature2': np.random.randn(100),
    'label': np.random.choice([0, 1], size=100)
})
print(df.head())

# Matplotlib: Simple plot
plt.scatter(df['feature1'], df['feature2'], c=df['label'])
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Feature Scatter Plot')
plt.show()

# Sklearn: Train/test split, model training, and evaluation
X = df[['feature1', 'feature2']]
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
NumPy array: [0 1 2 3 4 5 6 7 8 9]
   feature1  feature2  label
0  0.119612 -0.043873      1
1 -0.604721  0.100932      0
2  1.881212 -1.364462      1
3 -1.324307 -0.584629      0
4  0.829369 -1.275497      0
```

23°C Fog

Search

ENG IN

22:38 07-05-2025

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 2: Generate Sample Data (or use any dataset)
# Example: Predicting Salary based on Experience
data = {
    'Experience': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Salary': [30000, 35000, 40000, 45000, 50000, 55000, 60000, 65000, 70000, 75000]
}
df = pd.DataFrame(data)

# Step 3: Visualize Data
plt.scatter(df['Experience'], df['Salary'], color='blue')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Experience vs Salary')
plt.show()

# Step 4: Prepare Data
X = df[['Experience']]  # Independent variable
y = df['Salary']        # Dependent variable

# Step 5: Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Step 6: Train Model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 7: Predict
y_pred = model.predict(X_test)

# Step 8: Visualize Regression Line
plt.scatter(X_train, y_train, color='blue', label='Training data')
plt.plot(X_train, model.predict(X_train), color='red', linewidth=2, label='Regression line')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Linear Regression Fit')
```

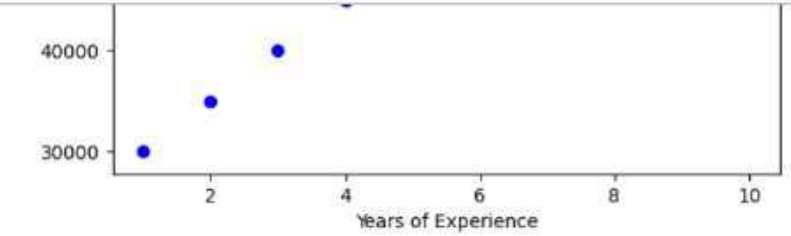
jupyter Untitled3 Last Checkpoint: 1 minute ago

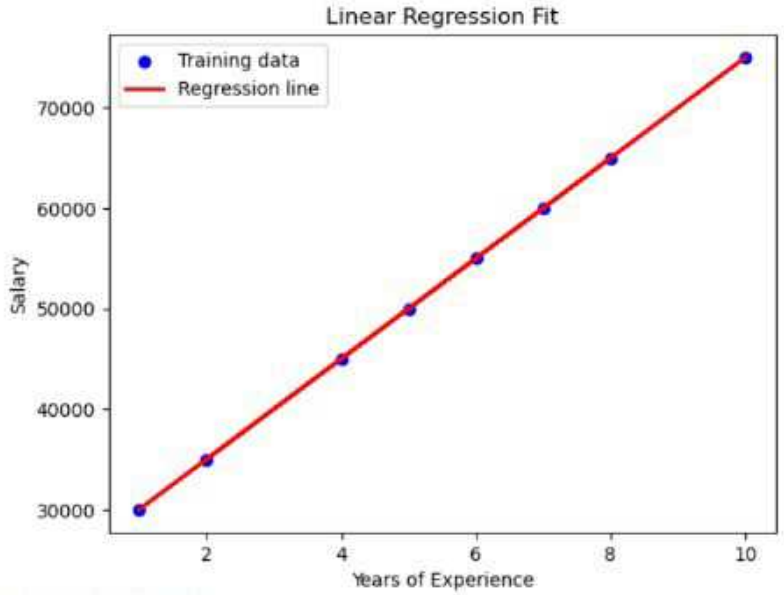
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python [conda env:base] *



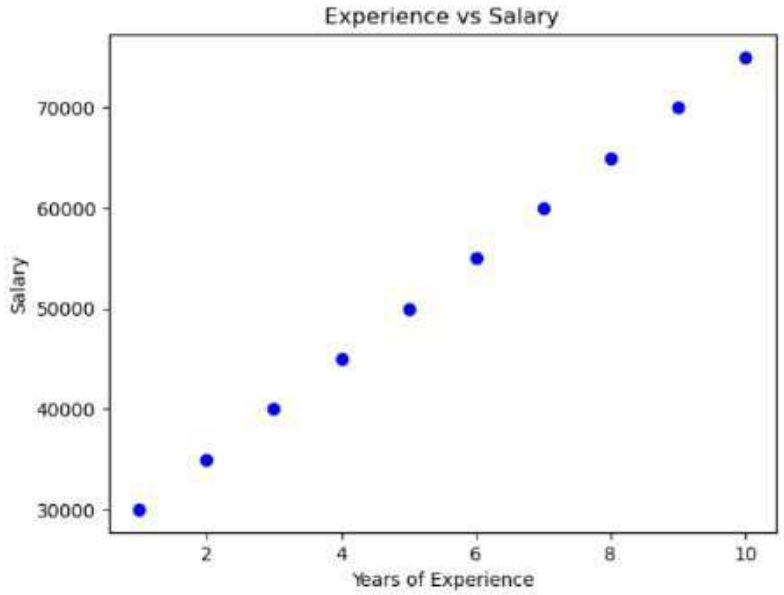



```
Coefficients: [5000.]
Intercept: 25000.0
Mean Squared Error (MSE): 0.0
R2 Score: 1.0
```

[ ]:

```
plt.legend()
plt.show()

# Step 9: Evaluation
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
```



localhost:8889/notebooks/Untitled4.ipynb?

jupyter Untitled4 Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python [conda env:base] * Anaconda Toolbox

[1]:

```
# 1. Importing Required Libraries
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn import tree
import matplotlib.pyplot as plt

# 2. Load Dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target

# Display first 5 rows
print("Iris Dataset:")
display(df.head())

# 3. Split Dataset
X = df.iloc[:, :-1]  # Features
y = df['target']    # Target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 4. Train Decision Tree Model
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# 5. Make Predictions
y_pred = clf.predict(X_test)

# 6. Evaluate the Model
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 7. Visualize the Decision Tree
plt.figure(figsize=(12, 8))
tree.plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names)
plt.title("Decision Tree Visualization")
plt.show()
```

Upcoming Earnings

Search

ENG IN

22:45 07-05-2025

Iris Dataset:

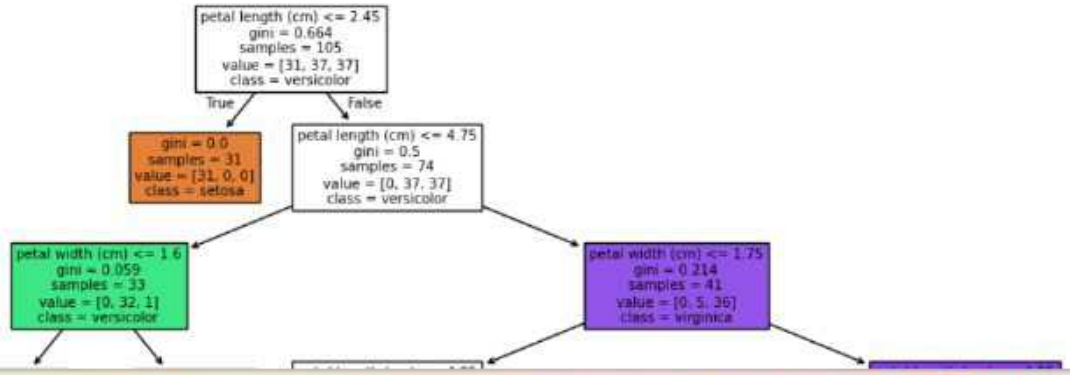
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Accuracy Score: 1.0

Classification Report:

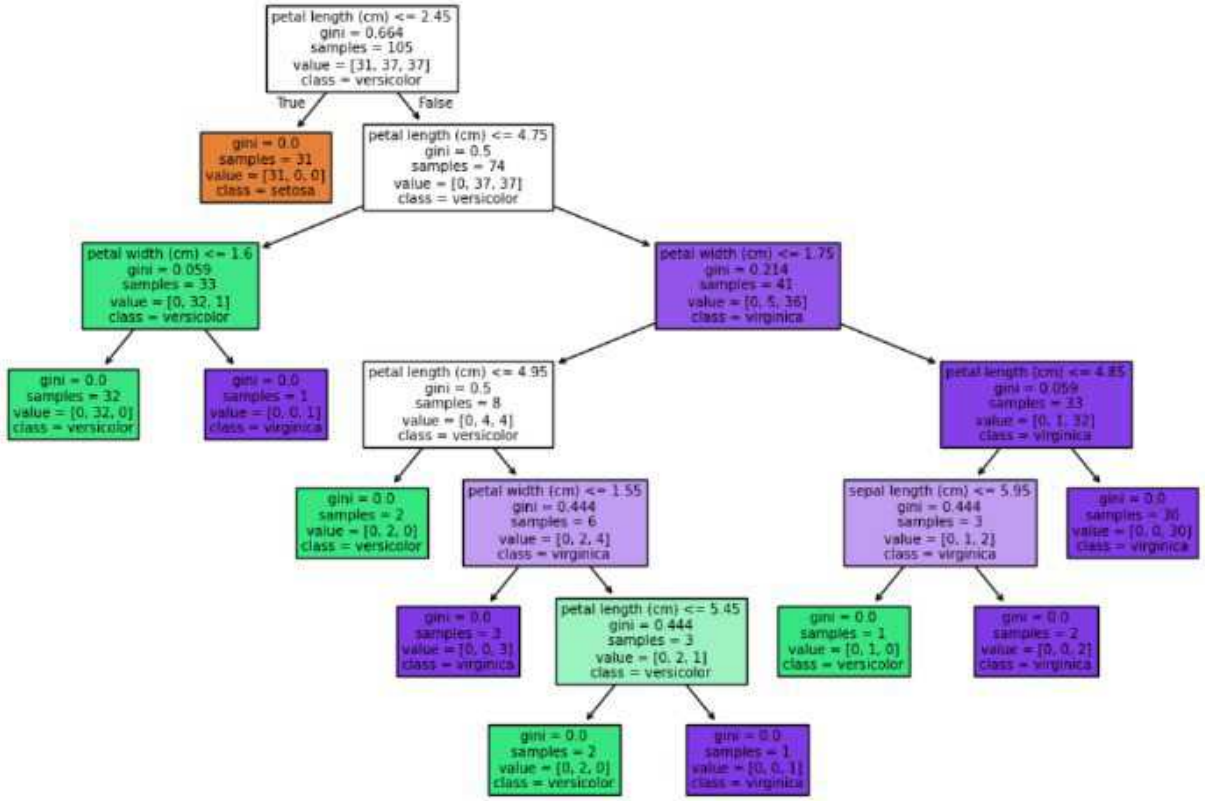
	precision	recall	f1-score	support
0	1.00	1.00	1.00	19
1	1.00	1.00	1.00	13
2	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

Decision Tree Visualization



accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

Decision Tree Visualization



[ ]: