

# DAY – 7

[11.02.2025]

Exp : 26 – 30

The image shows three screenshots of a Jupyter Notebook interface, each displaying a code cell and its output.

**Screenshot 1:** A code cell with ID [17] containing Python code for Logistic Regression. The code imports `LogisticRegression` from `sklearn.linear\_model`, uses `numpy` to create arrays `X` and `y`, fits a model, and prints the predicted value for a row with features 300 and 50. The output shows "Churn: 1".

```
[17]
from sklearn.linear_model import LogisticRegression
import numpy as np
X, y = np.array([[100,12],[300,36]]), np.array([0,1])
model = LogisticRegression().fit(X, y)
print("Churn:", model.predict([list(map(float,input().split()))])[0])
```

**Screenshot 2:** A code cell with ID [20] containing Python code for Linear Regression. The code imports `LinearRegression` from `sklearn.linear\_model`, uses `numpy` to create arrays `X` and `y`, fits a model, and prints the predicted price for a row with features 34.5 and 67.89. The output shows "Predicted Price: 1.725000000014416".

```
[20]
from sklearn.linear_model import LinearRegression
import numpy as np
X, y = np.array([[800,2],[1200,3],[1500,4]]), np.array([40,60,75])
model = LinearRegression().fit(X, y)
print("Predicted Price:", model.predict([list(map(float,input().split()))])[0])
```

**Screenshot 3:** A code cell with ID [22] containing Python code for Decision Tree Regression. The code imports `DecisionTreeRegressor` from `sklearn.tree`, uses `numpy` to create arrays `X` and `y`, fits a model, and prints the predicted price along with the decision tree's feature names and values. The output shows "Price: 10.0" and a detailed tree structure with nodes for Mileage and Age.

```
[22]
from sklearn.tree import DecisionTreeRegressor, export_text
import numpy as np
X, y = np.array([[50000,5],[20000,2],[70000,7]]), np.array([8,10,6])
m = DecisionTreeRegressor().fit(X, y)
print("Price:", m.predict([list(map(float,input().split()))])[0])
print(export_text(m, feature_names=["Mileage","Age"]))
```

The notebook interface includes a toolbar with file operations (File, Edit, View, Insert, Runtime, Tools, Help), a search bar, and a command bar with buttons for Commands, Code, Text, and Run all.

The image displays two side-by-side Jupyter Notebook interfaces, each showing a code cell and its output.

**Top Notebook (Untitled0.ipynb):**

- Cell 18:** [18] ✓ 0s
- Code:**

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import numpy as np
X, y = np.array([[1],[2],[3],[4]]), np.array([0,0,1,1])
m = LogisticRegression().fit(X, y); p = m.predict(X)
print(accuracy_score(y,p), precision_score(y,p), recall_score(y,p), f1_score(y,p))
```
- Output:**

```
... 1.0 1.0 1.0 1.0
```

**Bottom Notebook (Untitled0.ipynb):**

- Cell 19:** [19] ✓ 54s
- Code:**

```
from sklearn.cluster import KMeans
import numpy as np
X = np.array([[200,2],[800,8],[300,3],[900,9]])
model = KMeans(n_clusters=2).fit(X)
print("Segment:", model.predict([list(map(float,input().split()))])[0])
```
- Output:**

```
... 55 70
Segment: 1
```