ALFIDO TECH

TASK 2===

CODE=

Import required libraries

```
import pandas as pd

from sklearn.model_selection

import train_test_split

from sklearn.preprocessing

 import LabelEncoder, StandardScaler

from sklearn.linear_model

import LogisticRegression

 from sklearn.tree

 import DecisionTreeClassifier

from sklearn.metrics

import accuracy_score, confusion_matrix, classification_report
```

=== Step 1: Create sample data ===

```
data = { 'Name': ['Alice', 'Bob', 'Carol', 'David', 'Eva'],

 'Age': [25, 30, None, 35, 28],

 'Salary': [50000, None, 60000, 70000, 55000],

 'Gender': ['Female', 'Male', 'Female', 'Male', 'Female'],

 'Purchased': ['Yes', 'No', 'Yes', 'No', 'Yes'] }
```

```
df = pd.DataFrame(data)
```

=== Step 2: Preprocess ===

```
df['Age'].fillna(df['Age'].mean(), inplace=True)

df['Salary'].fillna(df['Salary'].mean(), inplace=True)
```

Encode categorical columns

```
label_enc = LabelEncoder()

for col in ['Name', 'Gender', 'Purchased']:

 df[col] = label_enc.fit_transform(df[col])
```

=== Step 3: Split into features (X) and target (y) ===

```
X = df.drop(columns=['Purchased'])

y = df['Purchased'] # keep as integer labels
```

=== Step 4: Scale features only (not the target!) ===

```
scaler = StandardScaler()

X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

=== Step 5: Train-test split ===

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

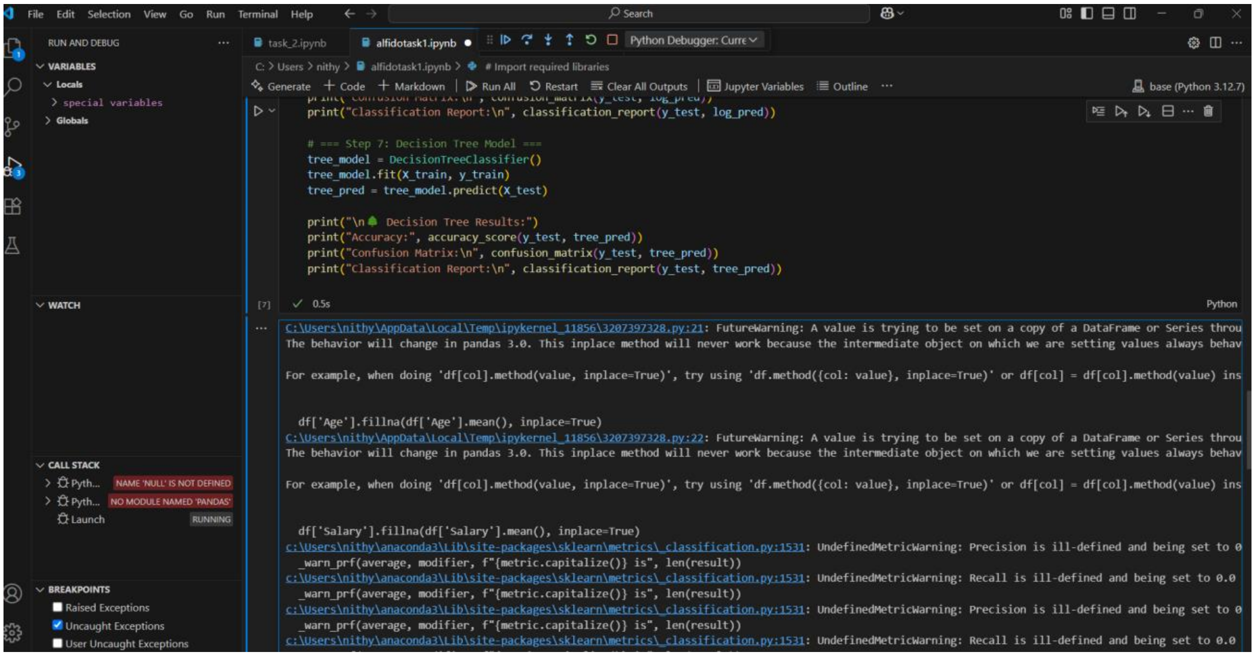
=== Step 6: Logistic Regression Model ===

```
log_model = LogisticRegression()

 log_model.fit(X_train, y_train)

log_pred = log_model.predict(X_test)

print("📈 Logistic Regression Results:")

 print("Accuracy:", accuracy_score(y_test, log_pred))

 print("Confusion Matrix:\n", confusion_matrix(y_test, log_pred))

print("Classification Report:\n", classification_report(y_test, log_pred))
```

=== Step 7: Decision Tree Model ===

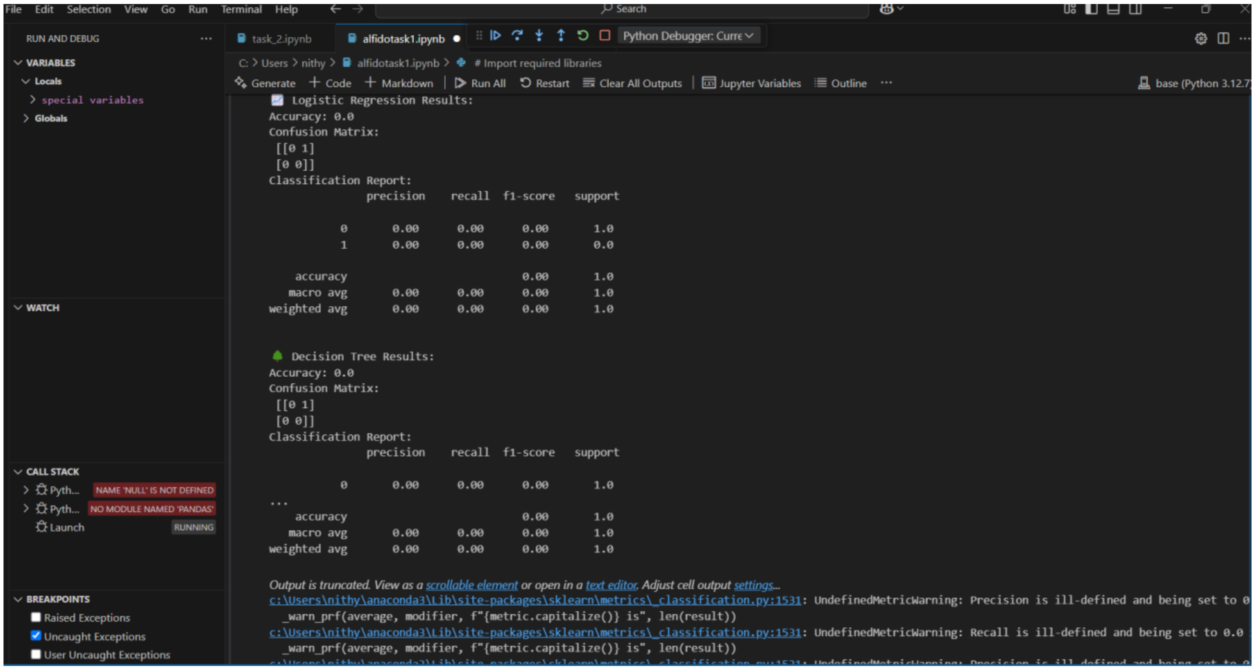
```
tree_model = DecisionTreeClassifier()

tree_model.fit(X_train, y_train)

tree_pred = tree_model.predict(X_test)

print("\n 🌳 Decision Tree Results:")

print("Accuracy:", accuracy_score(y_test, tree_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, tree_pred))

print("Classification Report:\n", classification_report(y_test, tree_pred))
```

OUTPUT====



```
print("Classification Report:\n", classification_report(y_test, log_pred))

# === Step 7: Decision Tree Model ===
tree_model = DecisionTreeClassifier()
tree_model.fit(X_train, y_train)
tree_pred = tree_model.predict(X_test)

print("\n🌲 Decision Tree Results:")
print("Accuracy:", accuracy_score(y_test, tree_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, tree_pred))
print("Classification Report:\n", classification_report(y_test, tree_pred))
```

```
C:\Users\nithy\AppData\Local\Temp\ipykernel_11856\3207397328.py:21: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series throu
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behav

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins


  df['Age'].fillna(df['Age'].mean(), inplace=True)
C:\Users\nithy\AppData\Local\Temp\ipykernel_11856\3207397328.py:22: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series throu
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behav

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins


  df['Salary'].fillna(df['Salary'].mean(), inplace=True)
c:\Users\nithy\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\nithy\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\nithy\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\nithy\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0
```



```
📈 Logistic Regression Results:
Accuracy: 0.0
Confusion Matrix:
 [[0 1]
 [0 0]]
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       0.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0


🌲 Decision Tree Results:
Accuracy: 0.0
Confusion Matrix:
 [[0 1]
 [0 0]]
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
...
    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
c:\Users\nithy\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\nithy\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```