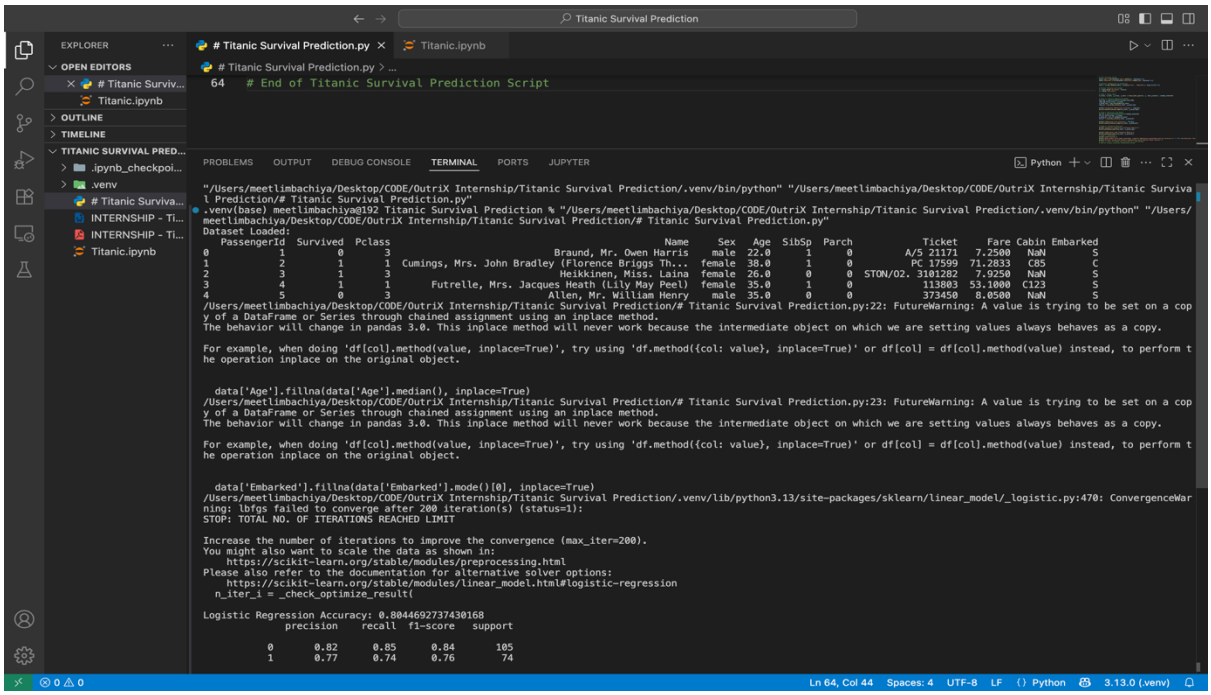
Titanic Survival Prediction : Internship

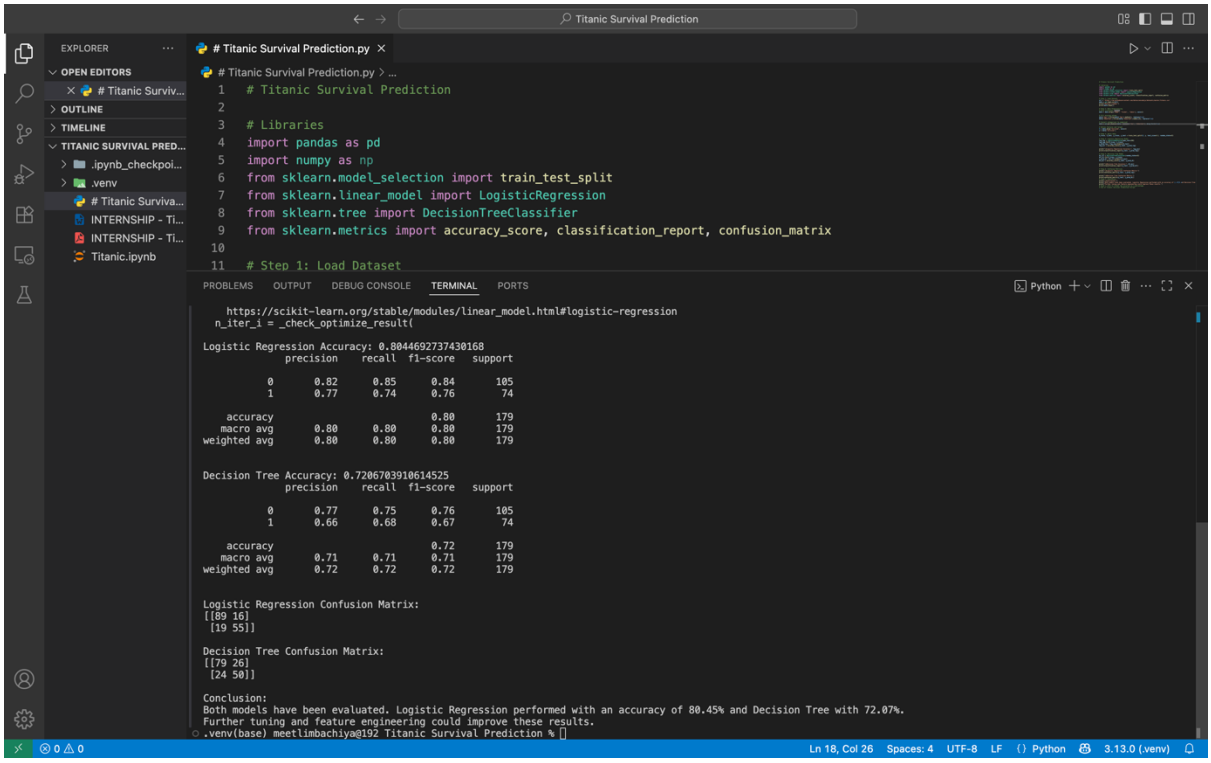
VS CODE:

```
# Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Step 1: Load Dataset
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
data = pd.read_csv(url)
print("Dataset Loaded:")
print(data.head())

# Step 2: Data Preprocessing
# Drop irrelevant columns
data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Fill missing values
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

# Convert categorical to numerical
data = pd.get_dummies(data, columns=['Sex', 'Embarked'], drop_first=True)

# Define features and target
X = data.drop('Survived', axis=1)
y = data['Survived']

# Step 3: Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Logistic Regression Model
log_reg = LogisticRegression(max_iter=200)
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)
log_acc = accuracy_score(y_test, y_pred_log)

print("\nLogistic Regression Accuracy:", log_acc)
print(classification_report(y_test, y_pred_log))

# Step 5: Decision Tree Model
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, y_train)
y_pred_dt = dt_clf.predict(X_test)
dt_acc = accuracy_score(y_test, y_pred_dt)

print("\nDecision Tree Accuracy:", dt_acc)
print(classification_report(y_test, y_pred_dt))
```

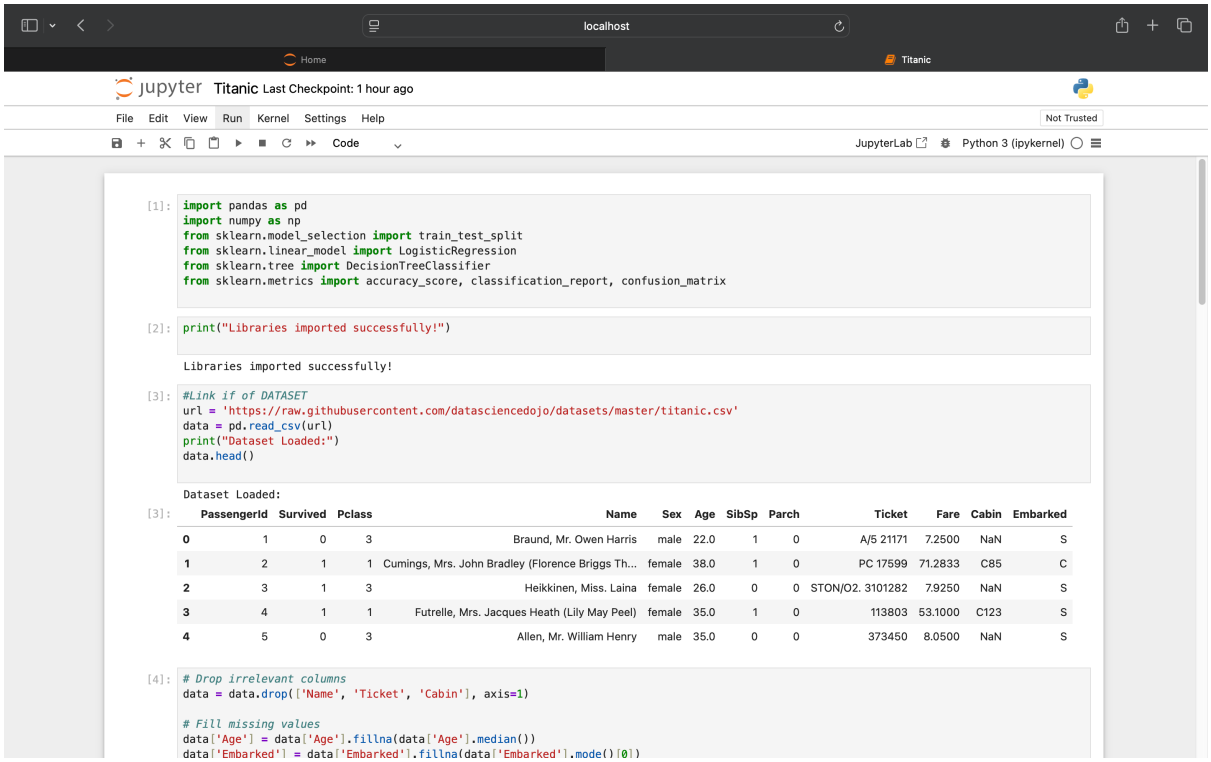
```
# Step 6: Confusion Matrices
print("\nLogistic Regression Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_log))
print("\nDecision Tree Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_dt))
# Step 7: Conclusion
print("\nConclusion:")
print("Both models have been evaluated. Logistic Regression performed with an accuracy of {:.2f}% and
Decision Tree with {:.2f}%.".format(log_acc * 100, dt_acc * 100))
print("Further tuning and feature engineering could improve these results.")
# Used {:.2f}% to format the accuracy as a percentage.
# End of Titanic Survival Prediction Script
```

OUTPUT:





JUPYTER NOTEBOOK:



[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

[2]:
```
print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

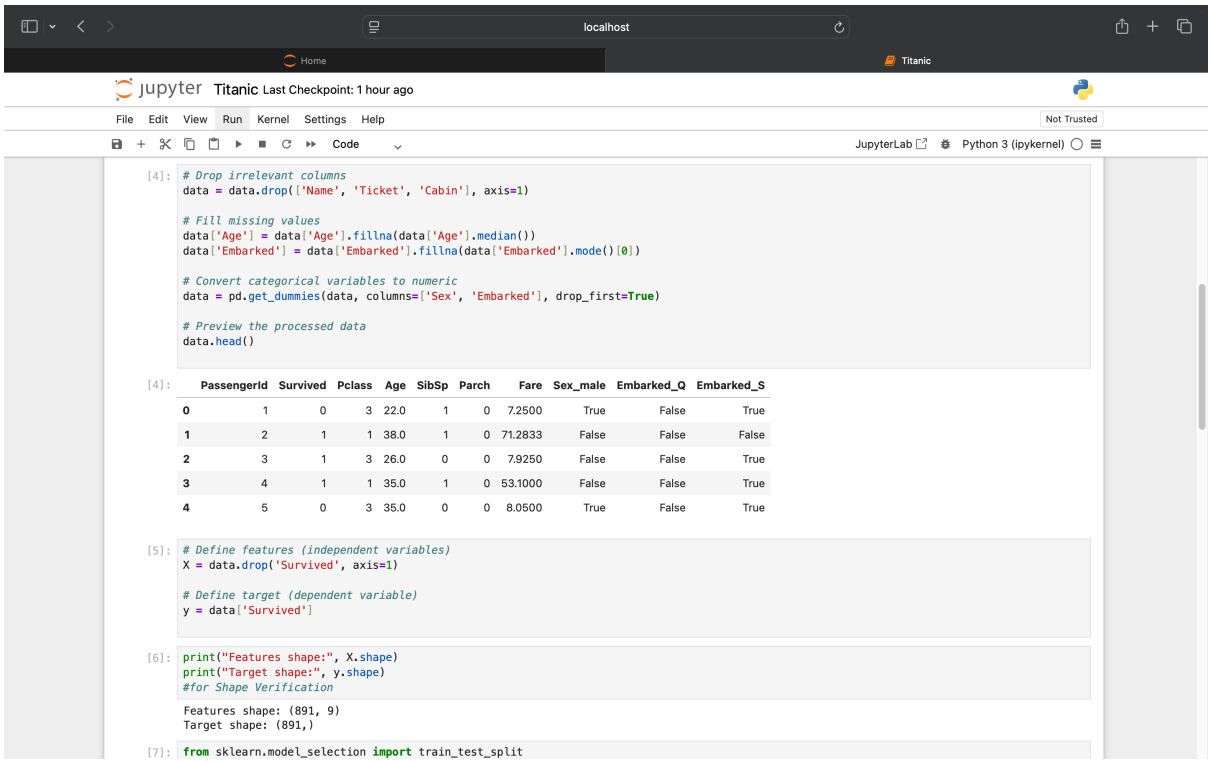
[3]:
```
#Link if of DATASET
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
data = pd.read_csv(url)
print("Dataset Loaded:")
data.head()
```

```
Dataset Loaded:
```

[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

[4]:
```
# Drop irrelevant columns
data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Fill missing values
data['Age'] = data['Age'].fillna(data['Age'].median())
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
```



[4]:
```
# Drop irrelevant columns
data = data.drop(['Name', 'Ticket', 'Cabin'], axis=1)

# Fill missing values
data['Age'] = data['Age'].fillna(data['Age'].median())
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])

# Convert categorical variables to numeric
data = pd.get_dummies(data, columns=['Sex', 'Embarked'], drop_first=True)

# Preview the processed data
data.head()
```

[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
0	1	0	3	22.0	1	0	7.2500	True	False	True
1	2	1	1	38.0	1	0	71.2833	False	False	False
2	3	1	3	26.0	0	0	7.9250	False	False	True
3	4	1	1	35.0	1	0	53.1000	False	False	True
4	5	0	3	35.0	0	0	8.0500	True	False	True

[5]:
```
# Define features (independent variables)
X = data.drop('Survived', axis=1)

# Define target (dependent variable)
y = data['Survived']
```

[6]:
```
print("Features shape:", X.shape)
print("Target shape:", y.shape)
#for Shape Verification
```

```
Features shape: (891, 9)
Target shape: (891,)
```

[7]:
```
from sklearn.model_selection import train_test_split
```

localhost

Home | Titanic

jupyter Titanic Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help — Not Trusted

Code — JupyterLab — Python 3 (ipykernel)

[7]:
```
from sklearn.model_selection import train_test_split

# For Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Training features shape:", X_train.shape)
print("Testing features shape:", X_test.shape)
print("Training target shape:", y_train.shape)
print("Testing target shape:", y_test.shape)
```

```
Training features shape: (712, 9)
Testing features shape: (179, 9)
Training target shape: (712,)
Testing target shape: (179,)
```

[8]:
```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train Logistic Regression on scaled data
log_reg = LogisticRegression(max_iter=500)
log_reg.fit(X_train_scaled, y_train)

# Predict on test data
y_pred_log = log_reg.predict(X_test_scaled)

# Evaluate performance
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_log))
print("Classification Report:\n", classification_report(y_test, y_pred_log))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log))

#0.804469 = 80%
```

```
Logistic Regression Accuracy: 0.8044692737430168
Classification Report:
```

localhost

Home | Titanic

jupyter Titanic Last Checkpoint: 1 hour ago

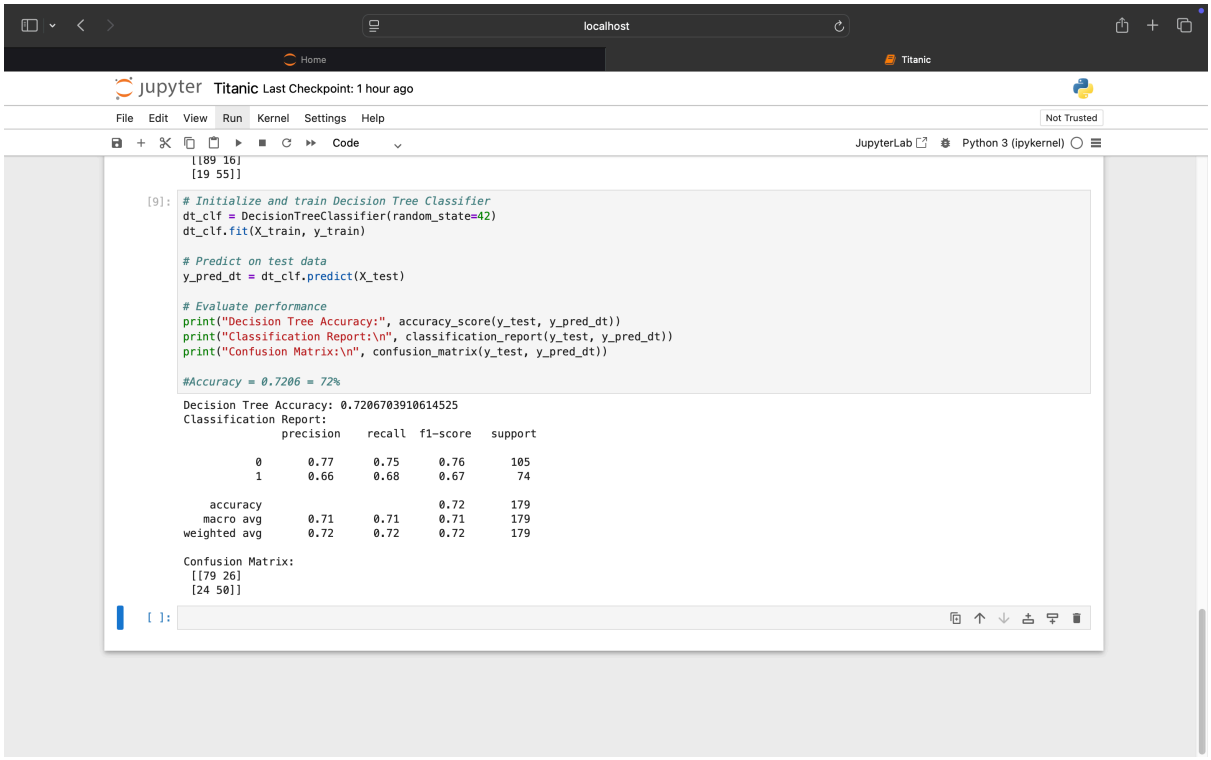
File Edit View Run Kernel Settings Help — Not Trusted

Code — JupyterLab — Python 3 (ipykernel)

```
Logistic Regression Accuracy: 0.8044692737430168
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.85      0.84       105
           1       0.77      0.74      0.76        74

    accuracy                           0.80       179
   macro avg       0.80      0.80      0.80       179
weighted avg       0.80      0.80      0.80       179

Confusion Matrix:
 [[89 16]
 [19 55]]
```

[9]:
```
# Initialize and train Decision Tree Classifier
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, y_train)

# Predict on test data
y_pred_dt = dt_clf.predict(X_test)

# Evaluate performance
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Classification Report:\n", classification_report(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))

#Accuracy = 0.7206 = 72%
```

```
Decision Tree Accuracy: 0.7206703910614525
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.75      0.76       105
           1       0.66      0.68      0.67        74

    accuracy                           0.72       179
   macro avg       0.71      0.71      0.71       179
weighted avg       0.72      0.72      0.72       179

Confusion Matrix:
```


localhost
Home
Titanic
jupyter Titanic Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)
[[89 16]
[19 55]]
[9]:
# Initialize and train Decision Tree Classifier
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, y_train)
# Predict on test data
y_pred_dt = dt_clf.predict(X_test)
# Evaluate performance
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Classification Report:\n", classification_report(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
#Accuracy = 0.7206 = 72%
Decision Tree Accuracy: 0.7206703910614525
Classification Report:
precision recall f1-score support
0 0.77 0.75 0.76 105
1 0.66 0.68 0.67 74
accuracy 0.72 179
macro avg 0.71 0.71 0.71 179
weighted avg 0.72 0.72 0.72 179
Confusion Matrix:
[[79 26]
[24 50]]
[ ]: