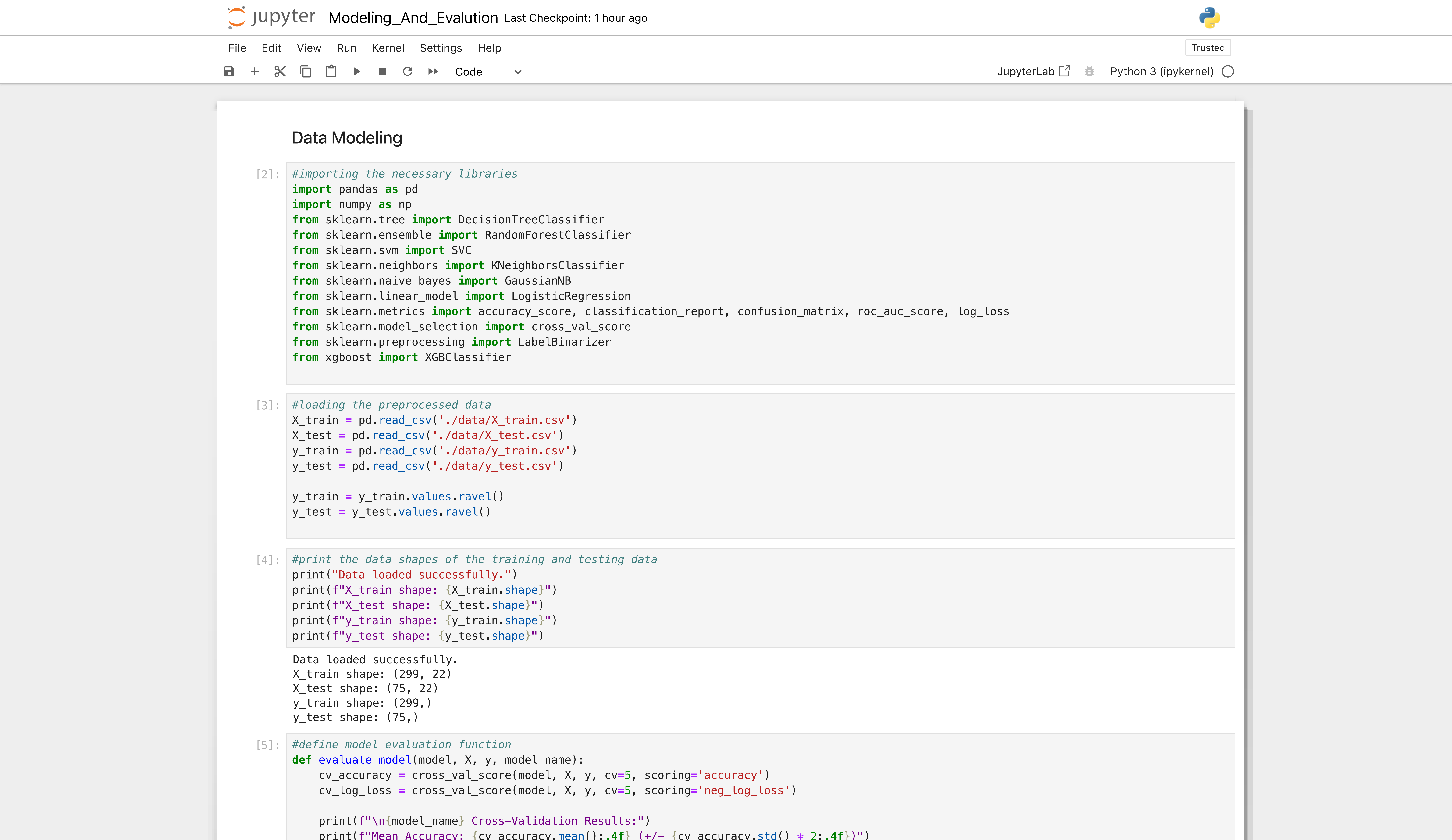

Data Modeling

[2]:

```
#importing the necessary libraries
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, log_loss
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelBinarizer
from xgboost import XGBClassifier
```

[3]:

```
#loading the preprocessed data
X_train = pd.read_csv('./data/X_train.csv')
X_test = pd.read_csv('./data/X_test.csv')
y_train = pd.read_csv('./data/y_train.csv')
y_test = pd.read_csv('./data/y_test.csv')

y_train = y_train.values.ravel()
y_test = y_test.values.ravel()
```

[4]:

```
#print the data shapes of the training and testing data
print("Data loaded successfully.")
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
Data loaded successfully.
X_train shape: (299, 22)
X_test shape: (75, 22)
y_train shape: (299,)
y_test shape: (75,)
```

[5]:

```
#define model evaluation function
def evaluate_model(model, X, y, model_name):
    cv_accuracy = cross_val_score(model, X, y, cv=5, scoring='accuracy')
    cv_log_loss = cross_val_score(model, X, y, cv=5, scoring='neg_log_loss')

    print(f"\n{model_name} Cross-Validation Results:")
    print(f"Mean Accuracy: {cv_accuracy.mean():.4f} (+/- {cv_accuracy.std() * 2:.4f})")
```