Project Title:

Prediction of ICICI Bank Stock Price based on Market Trends Using Machine Learning.

Software Testing Tool:

Software Testing tools are the tools that are used for the testing of software. Software testing tools are often used to assure firmness, thoroughness, and performance in testing software products. Unit testing and subsequent integration testing can be performed by software testing tools. These tools are used to fulfill all the requirements of planned testing activities. These tools also work as commercial software testing tools. The quality of the software is evaluated by software testers with the help of various testing tools.

Types of Testing Tools

Software testing is of two types, static testing, and dynamic testing. Also, the tools used during these tests are named accordingly on these testings. Testing tools can be categorized into two types which are as follows:

1. Static Test Tools: Static test tools work on the static testing processes. In the testing through these tools, the typical approach is taken. These tools do not test the real execution of the software. Certain input and output are not required in these tools. Static test tools consist of the following:

- **Flow analyzers:** Flow analyzers provide flexibility in the data flow from input to output.
- **Path Tests:** It finds the not used code and code with inconsistency in the software.
- **Coverage Analyzers:** All rationale paths in the software are assured by the coverage analyzers.
- **Interface Analyzers:** They check out the consequences of passing variables and data in the modules.

2. Dynamic Test Tools: The dynamic testing process is performed by the dynamic test tools. These tools test the software with existing or current data. Dynamic test tools comprise the following:

- **Test driver:** The test driver provides the input data to a module-under-test (MUT).
- **Test Beds:** It displays source code along with the program under execution at the same time.
- **Emulators:** Emulators provide the response facilities used to imitate parts of the system not yet developed.
- **Mutation Analyzers:** They are used for testing the fault tolerance of the system by knowingly providing the errors in the code of the software.

There is one more categorization of software testing tools. According to this classification, software testing tools are of 10 types:

1. **Test Management Tools:** Test management tools are used to store information on how testing is to be done, help to plan test activities, and report the status of quality assurance activities. For example, JIRA, Redmine, Selenium, etc.

2. **Automated Testing Tools:** Automated testing tools help to conduct testing activities without human intervention with more accuracy and less time and effort. For example, Appium, Cucumber, Ranorex, etc.
3. **Performance Testing Tools:** Performance testing tools help to perform effectively and efficiently performance testing which is a type of non-functional testing that checks the application for parameters like stability, scalability, performance, speed, etc. For example, WebLOAD, Apache JMeter, Neo Load, etc.
4. **Cross-browser Testing Tools:** Cross-browser testing tools help to perform cross-browser testing that lets the tester check whether the website works as intended when accessed through different browser-OS combinations. For example, Testsigma, Testim, Perfecto, etc.
5. **Integration Testing Tools:** Integration testing tools are used to test the interface between the modules and detect the bugs. The main purpose here is to check whether the specific modules are working as per the client's needs or not. For example, Citrus, FitNesse, TESSY, etc.
6. **Unit Testing Tools:** Unit testing tools are used to check the functionality of individual modules and to make sure that all independent modules work as expected. For example, Jenkins, PHPUnit, JUnit, etc.
7. **Mobile Testing Tools:** Mobile testing tools are used to test the application for compatibility on different mobile devices. For example, Appium, Robotium, Test IO, etc.
8. **GUI Testing Tools:** GUI testing tools are used to test the graphical user interface of the software. For example, EggPlant, Squish, AutoIT, etc.
9. **Bug Tracking Tools:** Bug tracking tool helps to keep track of various bugs that come up during the application lifecycle management. It helps to monitor and log all the bugs that are detected during software testing. For example, Trello, JIRA, GitHub, etc.
10. **Security Testing Tools:** Security testing is used to detect vulnerabilities and safeguard the application against malicious attacks. For example, NetSparker, Vega, ImmuniWeb, etc.

Software Testing Tool Platforms:

1. BrowserStack Test Management:

BrowserStack Test management is the latest software test management platform that offers a centralized test case repository with the best-in-class UI/UX. Integrates with other BrowserStack software testing tools such as Live, Test Observability, Automate & App Automate.

Features:

- Facilitates two-way integration with Jira, enhancing traceability for test cases and runs.
- Provides a rich dashboard for real-time reports & insights.
- Users can import data from existing tools using APIs or CSVs, with smart parsing for CSV fields.
- Test results can be uploaded from Test Observability or report formats like JUnit-XML/BDD-JSON.
- Supports test automation frameworks such as TestNG, WebdriverIO, Nightwatch.js, Appium, Playwright, etc.
- Integrates with CI/CD tools such as Jenkins, Azure Pipelines, Bamboo & CircleCI.

2. TestComplete:

TestComplete developed by SmartBear Software is a functional automated testing tool that ensures the quality of the application without sacrificing quality or agility.

Features:

- TestComplete has built-in keyword-driven test editor that consists of keyword operations that correspond to automated testing actions.
- It records the key actions that are necessary to replay test and discard all unneeded actions.
- It can run several automated tests across separate virtual machines.
- It has built-in code editor that helps testers write scripts manually.

- It automatically captures screenshots during test recording and playback.

3. LambdaTest:

LambdaTest is a cross-browser testing tool that helps to evaluate how web application responds when accessed through a variety of different browsers.

Features:

- It has Selenium scripts on 3000+ browsers and operating system environments, giving higher test coverage.
- It can perform automated cross-browser testing of locally hosted web pages using LambdaTest tunnel.
- It can also help to run a single test across multiple browser/ OS configurations simultaneously.

4. TestRail:

TestRail is a test management tool that helps to streamline software testing processes, get visibility into QA. This tool is used by testers, developers, and team leads to manage, track, and organize software testing efforts.

Features:

- It helps to manage test cases, plans, and runs.
- It helps to increase test coverage.
- It helps to get real-time insights into your QA progress.
- It helps to document test plans and track real-time progress.

5. Xray:

Xray is a test management app for Jira that helps to plan, execute, and track quality assurance with requirements traceability.

Features:

- It promotes Native Quality Management, where all tools, tests used by QA are built natively into development environment like Jira.

- It integrates with leading automation frameworks like Cucumber, Selenium, and JUnit to automate testing.
- It allows easy integration with CI tools like Jenkins, Bamboo, and GitLab.
- It helps to easily map stories using BDD.

ML Model Application Program:

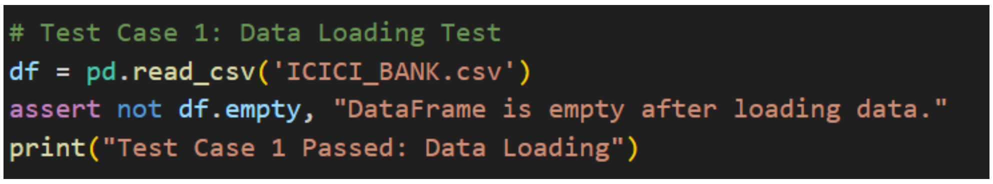
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics
from sklearn.impute import SimpleImputer
from sklearn.metrics import confusion_matrix  # Import confusion_matrix from
sklearn.metrics

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('ICICI_BANK.csv')
assert not df.empty, "DataFrame is empty after loading data."

assert df.shape == (len(df), 7), "DataFrame shape is incorrect."
assert df['Close'].dtype == np.float64, "Close column type is incorrect."

plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('ICICI Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()

features = ['Open', 'High', 'Low', 'Close', 'Volume']
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2, 3, i+1)
    sb.distplot(df[col])
plt.show()
```

```python
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')
df['day'] = df['Date'].dt.day
df['month'] = df['Date'].dt.month
df['year'] = df['Date'].dt.year
assert 'day' in df.columns and 'month' in df.columns and 'year' in df.columns,
"Date extraction failed."

df['is_quarter_end'] = (df['month'] % 3 == 0).astype(int)
assert 'is_quarter_end' in df.columns, "Quarter-end indicator creation
failed."

df['target'] = (df['Close'].shift(-1) > df['Close']).astype(int)
assert 'target' in df.columns, "Target variable creation failed."

features = df[['Open', 'High', 'Low', 'Close', 'Volume', 'day', 'month',
'year', 'is_quarter_end']]
target = df['target']

imputer = SimpleImputer(strategy='mean')
features_imputed = imputer.fit_transform(features)

scaler = StandardScaler()
features_scaled = scaler.fit_transform(features_imputed)
X_train, X_valid, Y_train, Y_valid = train_test_split(features_scaled, target,
test_size=0.1, random_state=2022)
assert X_train.shape[0] > 0 and X_valid.shape[0] > 0, "Data split for
training/validation failed."

models = [
    LogisticRegression(),
    SVC(kernel='poly', probability=True),
    XGBClassifier()
]

for model in models:
    model.fit(X_train, Y_train)
    train_auc = metrics.roc_auc_score(Y_train, model.predict_proba(X_train)[:,
1])
    valid_auc = metrics.roc_auc_score(Y_valid, model.predict_proba(X_valid)[:,
1])
    assert train_auc > 0.5 and valid_auc > 0.5, f"Model {type(model).__name__}
evaluation failed."

plt.pie(df['target'].value_counts().values, labels=[0, 1], autopct='%1.1f%%')
plt.show()

plt.figure(figsize=(10, 10))
sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
```

```
plt.show()

cm = confusion_matrix(Y_valid, models[0].predict(X_valid))
sb.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Test cases:

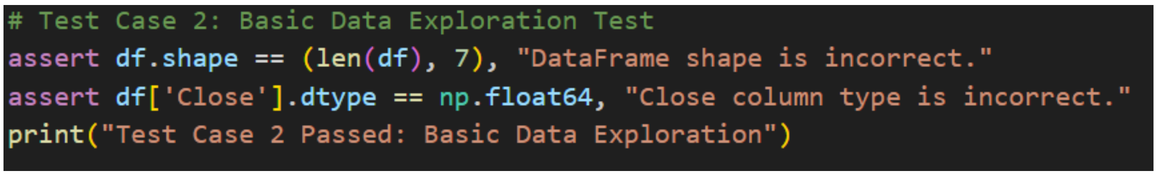
Test case 1: Data loading test



```
# Test Case 1: Data Loading Test
df = pd.read_csv('ICICI_BANK.csv')
assert not df.empty, "DataFrame is empty after loading data."
print("Test Case 1 Passed: Data Loading")
```

Reading the data file of ICICI stock price.

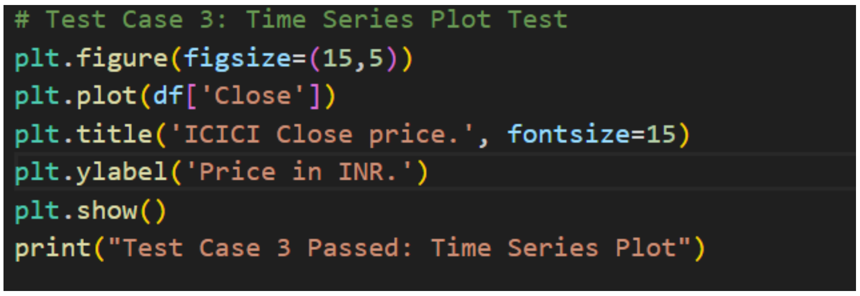
Test case 2: Basic data exploration test



```
# Test Case 2: Basic Data Exploration Test
assert df.shape == (len(df), 7), "DataFrame shape is incorrect."
assert df['Close'].dtype == np.float64, "Close column type is incorrect."
print("Test Case 2 Passed: Basic Data Exploration")
```

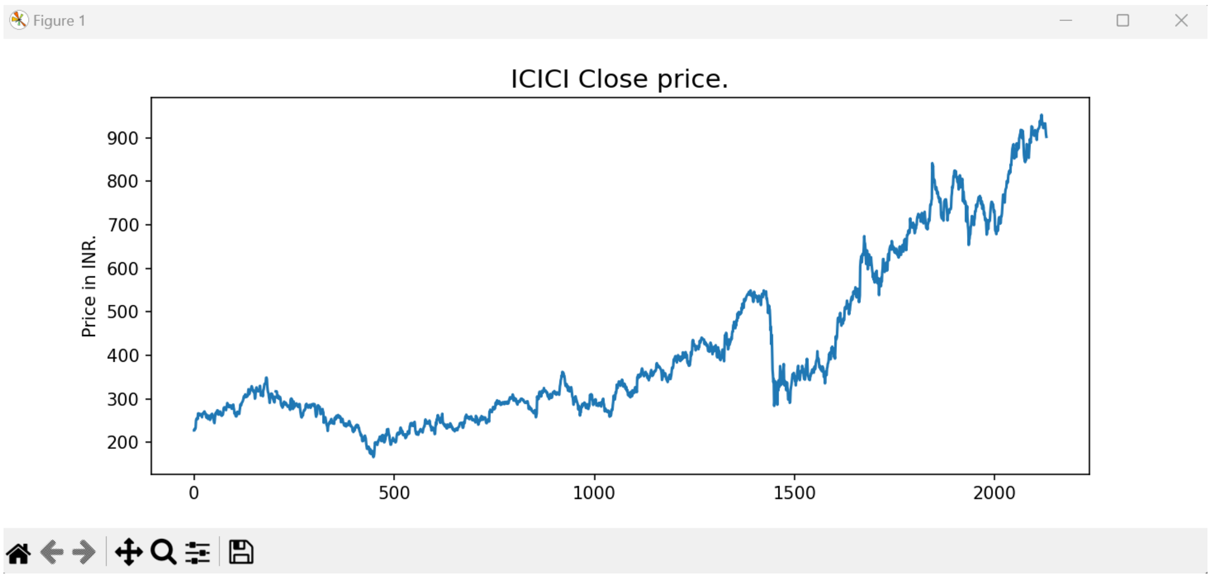
Data frame of the stock data file is explored and tested for format.

Test case 3: Time series plot test



```
# Test Case 3: Time Series Plot Test
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('ICICI Close price.', fontsize=15)
plt.ylabel('Price in INR.')
plt.show()
print("Test Case 3 Passed: Time Series Plot")
```

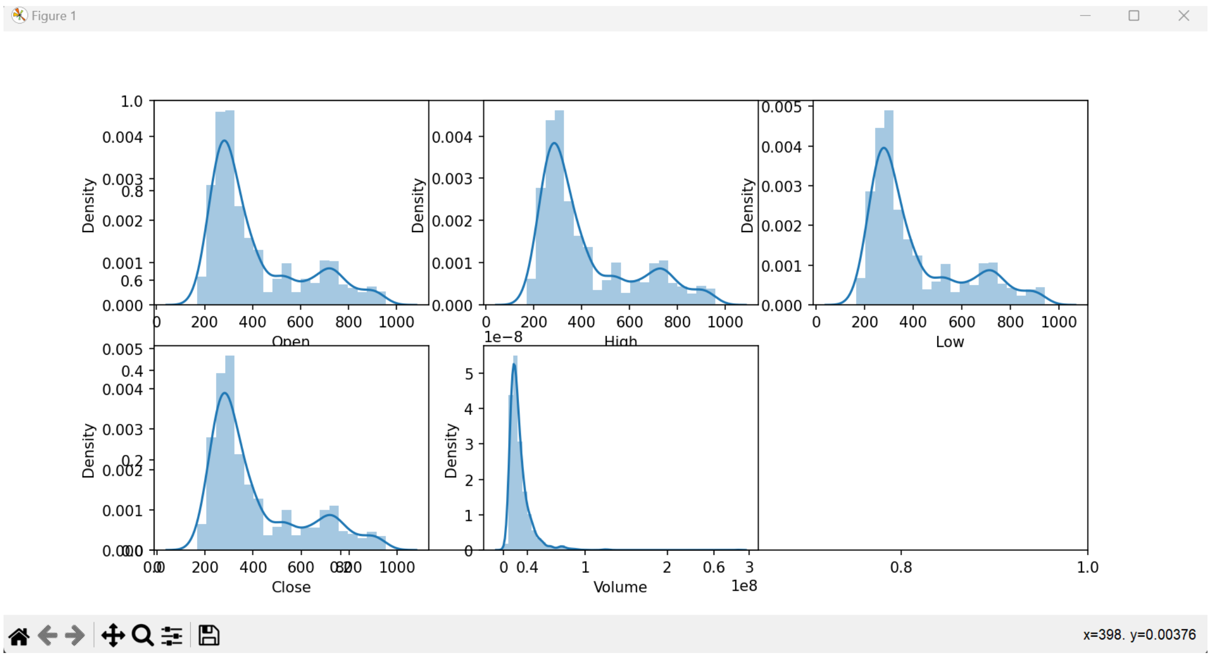
Closing price graph of ICICI stock price is displayed.



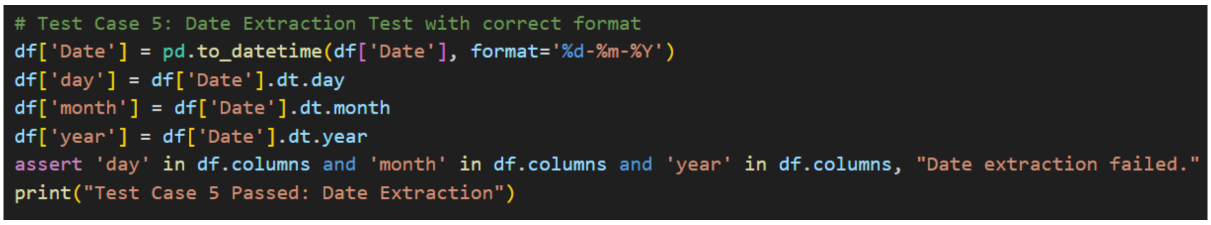
Test case 4: Feature distribution visualization test

```
# Test Case 4: Feature Distribution Visualization Test
features = ['Open', 'High', 'Low', 'Close', 'Volume']
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2, 3, i+1)
    sb.~~distplot~~(df[col])
plt.show()
print("Test Case 4 Passed: Feature Distribution Visualization")
```

A graph of stock price factors : (open, high, low, close, volume) is shown.



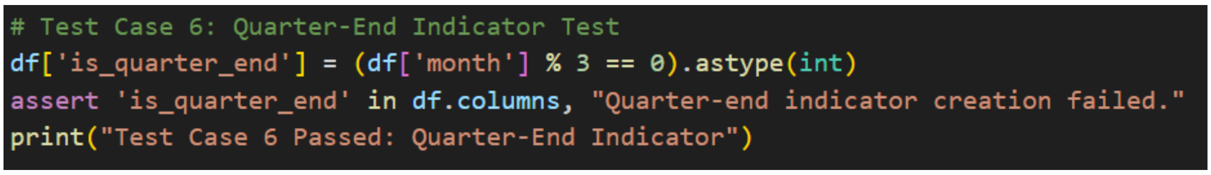
Test case 5: Data extraction test with correct format



```python
# Test Case 5: Date Extraction Test with correct format
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')
df['day'] = df['Date'].dt.day
df['month'] = df['Date'].dt.month
df['year'] = df['Date'].dt.year
assert 'day' in df.columns and 'month' in df.columns and 'year' in df.columns, "Date extraction failed."
print("Test Case 5 Passed: Date Extraction")
```

Day, month, and year are extracted for analysis.

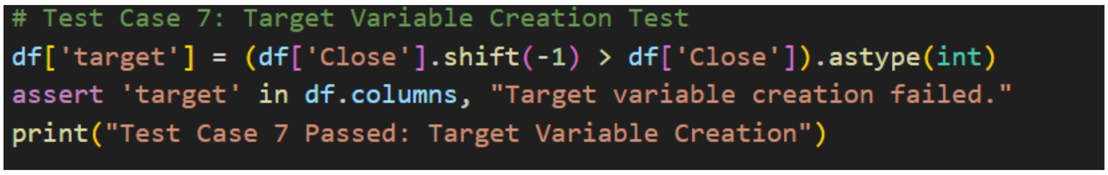
Test case 6: Quarter-end indicator test



```python
# Test Case 6: Quarter-End Indicator Test
df['is_quarter_end'] = (df['month'] % 3 == 0).astype(int)
assert 'is_quarter_end' in df.columns, "Quarter-end indicator creation failed."
print("Test Case 6 Passed: Quarter-End Indicator")
```

Quarter-end indicator of the year is verified.

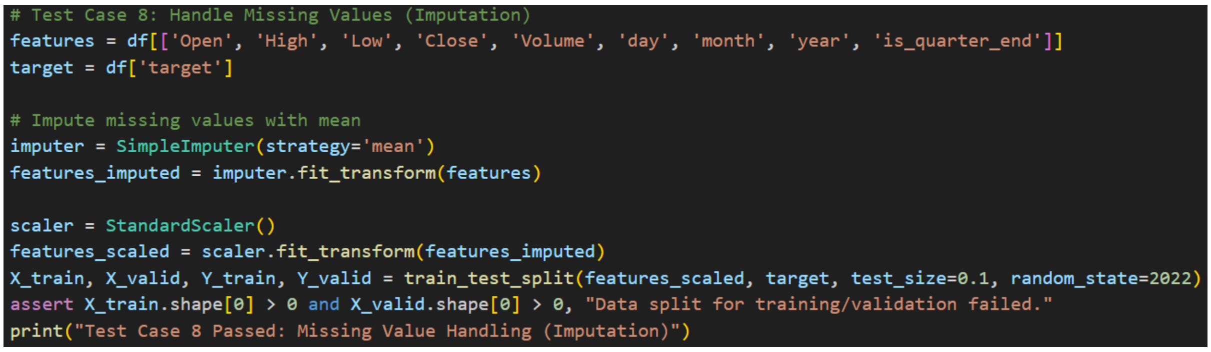
Test case 7: Target variable creation test



```python
# Test Case 7: Target Variable Creation Test
df['target'] = (df['Close'].shift(-1) > df['Close']).astype(int)
assert 'target' in df.columns, "Target variable creation failed."
print("Test Case 7 Passed: Target Variable Creation")
```

Target variables (stock close price) is verified.

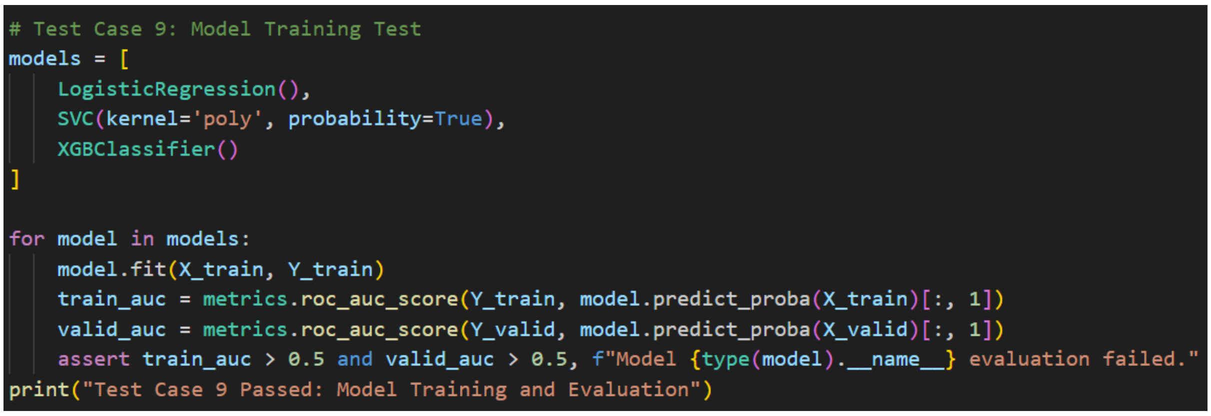
Test case 8: Handle missing value (imputation) test



```python
# Test Case 8: Handle Missing Values (Imputation)
features = df[['Open', 'High', 'Low', 'Close', 'Volume', 'day', 'month', 'year', 'is_quarter_end']]
target = df['target']

# Impute missing values with mean
imputer = SimpleImputer(strategy='mean')
features_imputed = imputer.fit_transform(features)

scaler = StandardScaler()
features_scaled = scaler.fit_transform(features_imputed)
X_train, X_valid, Y_train, Y_valid = train_test_split(features_scaled, target, test_size=0.1, random_state=2022)
assert X_train.shape[0] > 0 and X_valid.shape[0] > 0, "Data split for training/validation failed."
print("Test Case 8 Passed: Missing Value Handling (Imputation)")
```

Any missing NaN values in the data set is handled.

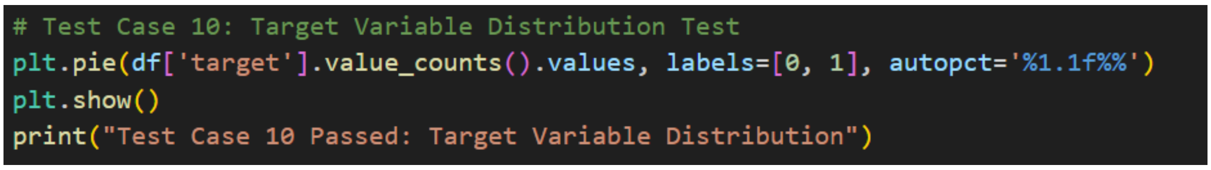
Test case 9: Model training test



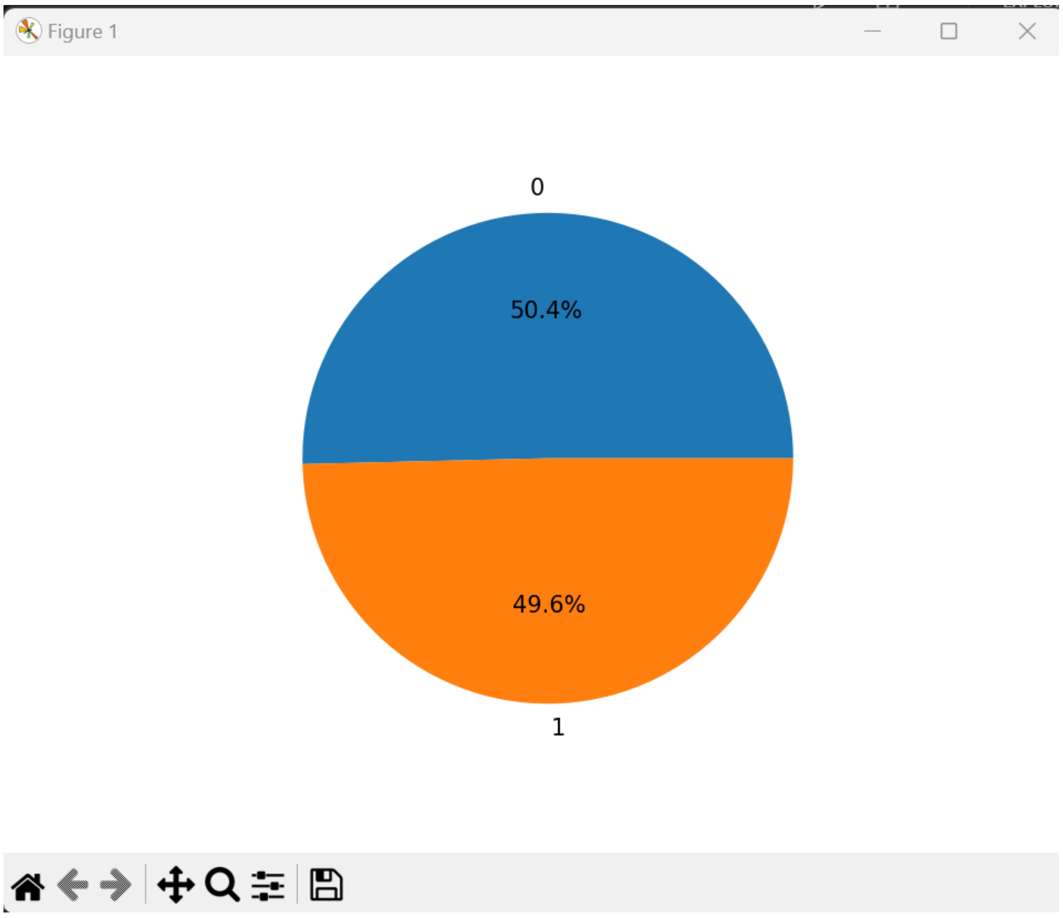
```
# Test Case 9: Model Training Test
models = [
    LogisticRegression(),
    SVC(kernel='poly', probability=True),
    XGBClassifier()
]

for model in models:
    model.fit(X_train, Y_train)
    train_auc = metrics.roc_auc_score(Y_train, model.predict_proba(X_train)[:, 1])
    valid_auc = metrics.roc_auc_score(Y_valid, model.predict_proba(X_valid)[:, 1])
    assert train_auc > 0.5 and valid_auc > 0.5, f"Model {type(model).__name__} evaluation failed."
print("Test Case 9 Passed: Model Training and Evaluation")
```

Model is tested and verified with predictions output.

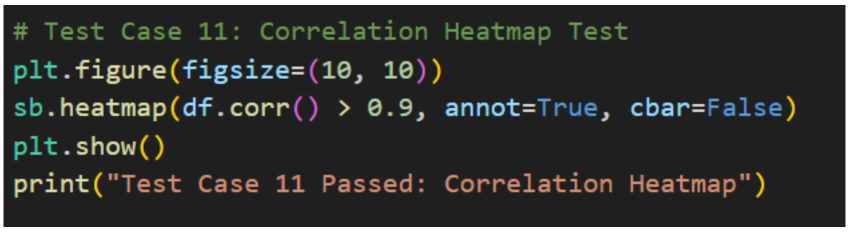
Test case 10: Target variable distribution test



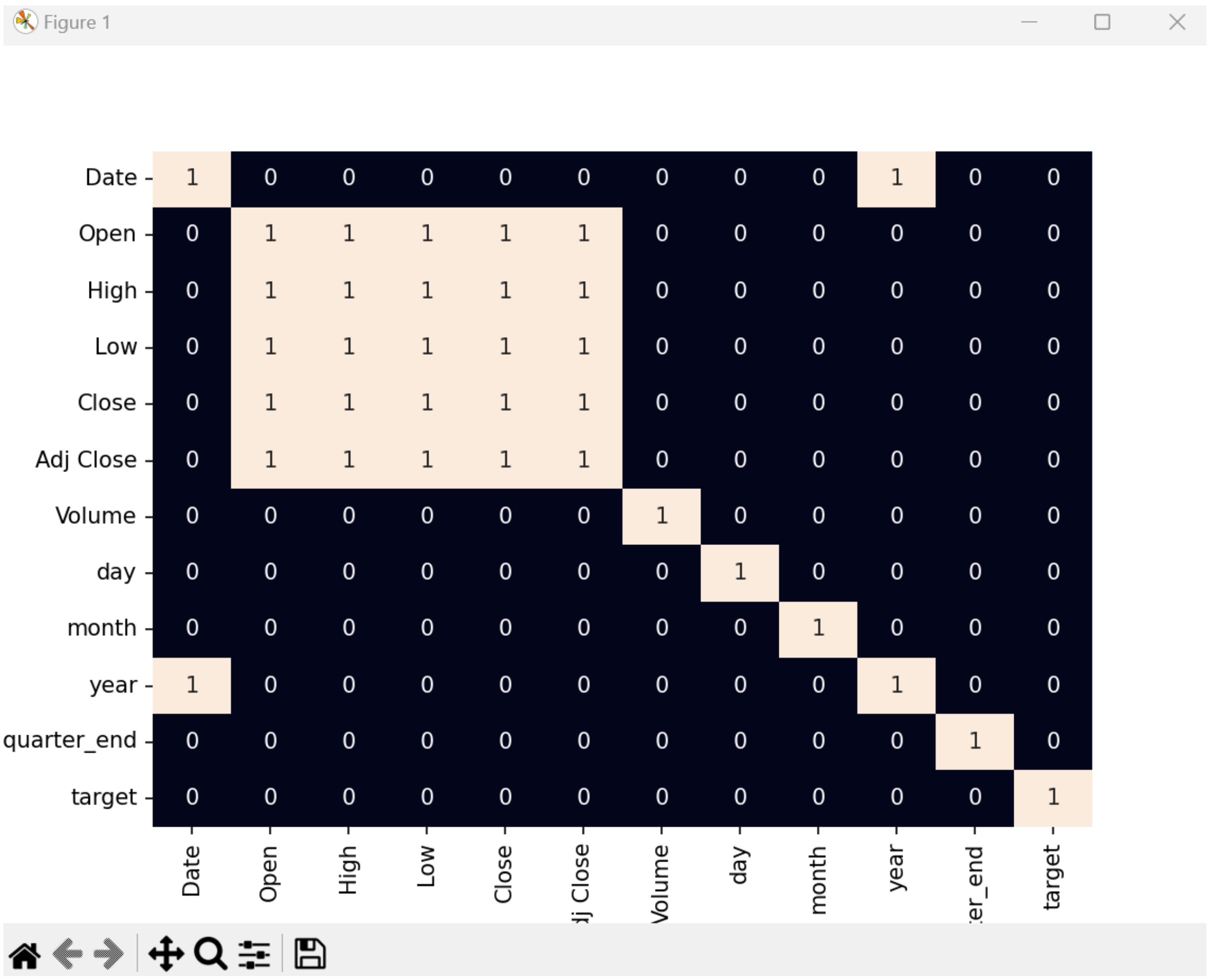
```
# Test Case 10: Target Variable Distribution Test
plt.pie(df['target'].value_counts().values, labels=[0, 1], autopct='%1.1f%%')
plt.show()
print("Test Case 10 Passed: Target Variable Distribution")
```



Test case 11: Correlation heatmap test



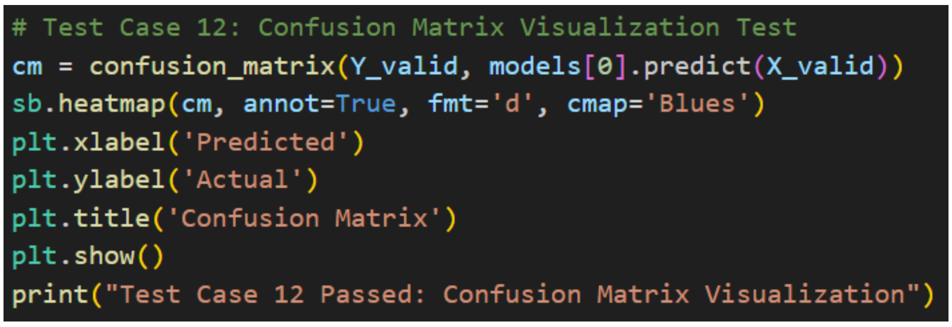
```python
# Test Case 11: Correlation Heatmap Test
plt.figure(figsize=(10, 10))
sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
plt.show()
print("Test Case 11 Passed: Correlation Heatmap")
```



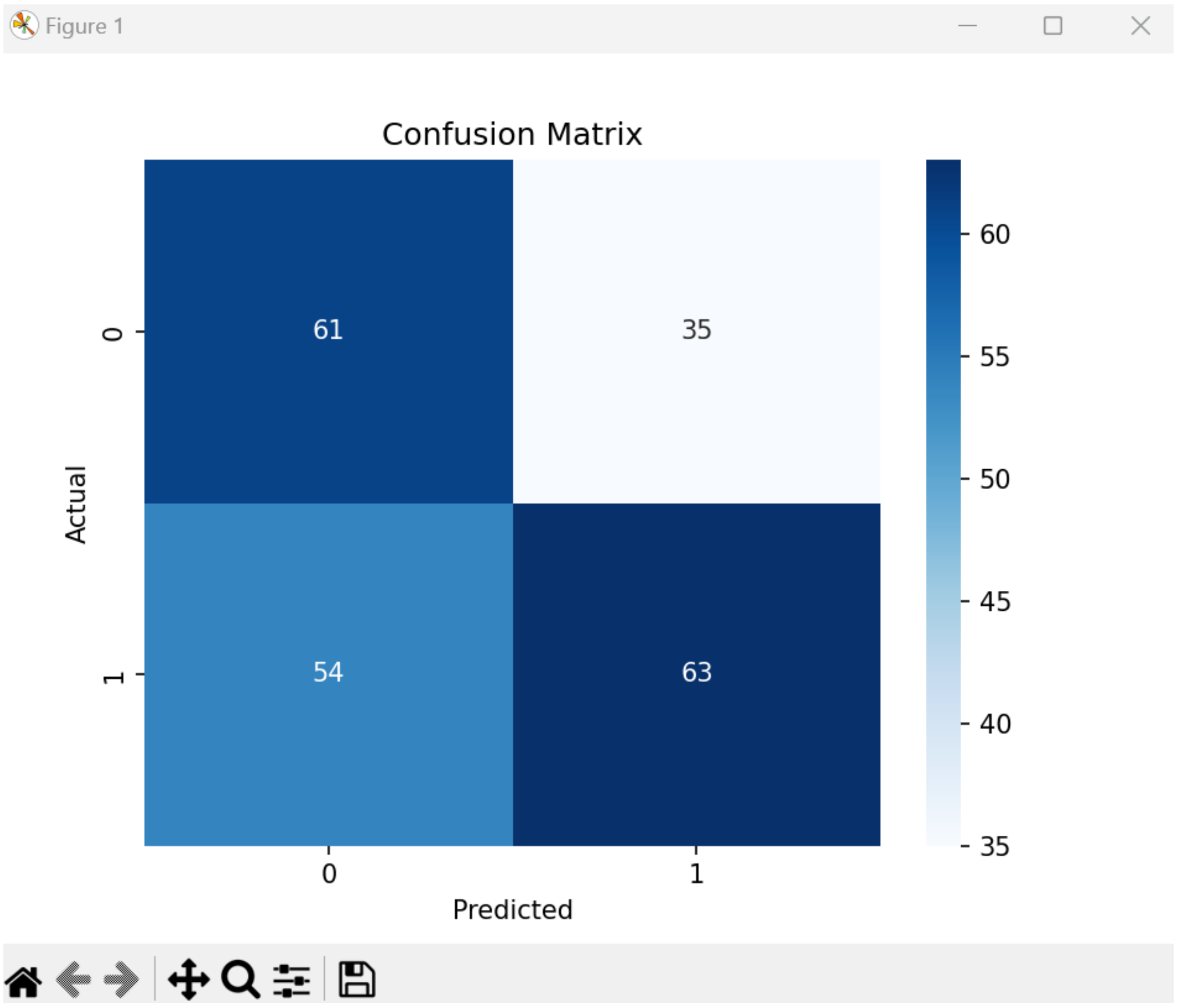
	Date	Open	High	Low	Close	dj Close	Volume	day	month	year	er_end	target
Date	1	0	0	0	0	0	0	0	0	1	0	0
Open	0	1	1	1	1	1	0	0	0	0	0	0
High	0	1	1	1	1	1	0	0	0	0	0	0
Low	0	1	1	1	1	1	0	0	0	0	0	0
Close	0	1	1	1	1	1	0	0	0	0	0	0
Adj Close	0	1	1	1	1	1	0	0	0	0	0	0
Volume	0	0	0	0	0	0	1	0	0	0	0	0
day	0	0	0	0	0	0	0	1	0	0	0	0
month	0	0	0	0	0	0	0	0	1	0	0	0
year	1	0	0	0	0	0	0	0	0	1	0	0
quarter_end	0	0	0	0	0	0	0	0	0	0	1	0
target	0	0	0	0	0	0	0	0	0	0	0	1

A correlation heatmap matrix is produced by analyzing the correlation coefficient.

Test case 12: Confusion matrix visualization test



```
# Test Case 12: Confusion Matrix Visualization Test
cm = confusion_matrix(Y_valid, models[0].predict(X_valid))
sb.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
print("Test Case 12 Passed: Confusion Matrix Visualization")
```



ICICI stock price data set:

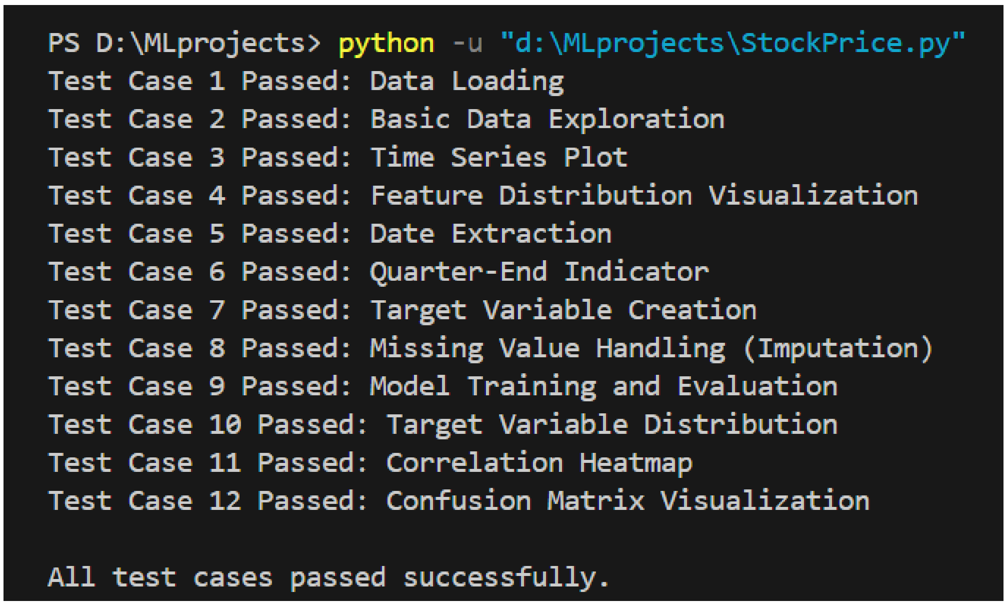
```
Date,Open,High,Low,Close,Adj Close,Volume
02-05-2014,227.27272,229.854538,226.736359,227.709091,204.749756,9145994
05-05-2014,226.590912,232.181824,225.090912,227.718185,204.75795,14052241
06-05-2014,229.854538,232.727264,229.854538,231.927277,208.542603,7045379
07-05-2014,232,232.181824,228.490906,231.027267,207.733383,28329075
08-05-2014,231.800003,235.781815,230.227264,234.363632,210.733337,16439945
09-05-2014,233.663635,252.890915,233.090912,250.654541,225.381683,34424511
12-05-2014,251.281815,257.145447,247.300003,254.309097,228.66774,28438118
13-05-2014,257.818176,259.636353,253.363632,254.690903,229.011108,22394201
14-05-2014,252.181824,257.018188,250.399994,256.254547,230.417053,20961468
15-05-2014,256.254547,256.345459,251.845459,253.654541,228.079208,22401164
16-05-2014,263.636353,289.672729,257.763641,266.418182,239.555939,64509505
19-05-2014,269.272736,271.554535,262.772736,266.981812,240.062729,42620820
20-05-2014,269.081818,270,261.909088,263.618195,237.038284,22394427
21-05-2014,264.545441,264.899994,258.381805,261.918182,235.509644,19602885
22-05-2014,262.200012,270,262.200012,263.909088,237.299835,17050280
23-05-2014,265.481812,267.227264,262.727264,265.600006,238.820251,18774596
26-05-2014,265.090912,272.381805,262.727264,264.463623,237.798447,22024612
27-05-2014,262.727264,264.654541,259.554535,263.1091,236.58046,10500330
28-05-2014,263.645447,267.054535,262.054535,264.763641,238.068176,13082932
29-05-2014,265.090912,267.181824,259.418182,261.263641,234.921127,24559144
30-05-2014,261.909088,263.345459,255.409088,257.827271,231.831192,17143027
02-06-2014,256.590912,267.272736,256.381805,266.21817,239.376099,15135730
03-06-2014,265.854553,268.672729,261.827271,265.154541,238.419678,19736579
04-06-2014,265.436371,270.28183,262.545441,267.163635,244.075577,12880208
05-06-2014,263.090912,267.454559,261.336365,266.027283,246.902115,21069081
06-06-2014,267.745453,273.336365,265.454559,270.236359,250.808594,21390919
09-06-2014,270.209076,274.854553,269.727264,271.309082,251.804169,21202940
10-06-2014,271.363647,271.363647,268,269.318176,249.956467,10364431
11-06-2014,268.727264,273.1091,265.454559,267.254547,248.041153,12749231
12-06-2014,267.272736,268,263.299988,265.345459,246.269287,11788678
13-06-2014,265.872742,268.909088,258.472717,259.700012,241.029739,16782672
16-06-2014,258.836365,259.818176,254.727264,257.081818,238.599747,13089175
17-06-2014,257.090912,263.972717,255.818176,263.545441,244.598694,15687243
18-06-2014,262.881805,265.045441,256.772736,258.454559,239.873825,13210807
19-06-2014,260.436371,261.009094,254.045456,256.681824,238.2285,16235054
20-06-2014,257.272736,259.090912,253.181824,253.972733,235.714203,16995341
23-06-2014,254.545456,258.090912,254.081818,257.127258,238.641922,11711777
24-06-2014,259.272736,262.8909,257.581818,261.690918,242.877487,12547546
```

:

:

Stock price data from 2014 to 2022

Test cases passed verification:



```
PS D:\MLprojects> python -u "d:\MLprojects\StockPrice.py"
Test Case 1 Passed: Data Loading
Test Case 2 Passed: Basic Data Exploration
Test Case 3 Passed: Time Series Plot
Test Case 4 Passed: Feature Distribution Visualization
Test Case 5 Passed: Date Extraction
Test Case 6 Passed: Quarter-End Indicator
Test Case 7 Passed: Target Variable Creation
Test Case 8 Passed: Missing Value Handling (Imputation)
Test Case 9 Passed: Model Training and Evaluation
Test Case 10 Passed: Target Variable Distribution
Test Case 11 Passed: Correlation Heatmap
Test Case 12 Passed: Confusion Matrix Visualization

All test cases passed successfully.
```

Hence, the model is tested with 12 different phases of test cases to verify the efficiency and working of all the features as per client specifications.