DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Code:

```
from datetime import datetime

class User:
    def __init__(self, user_id, balance):
        self.user_id = user_id
        self.balance = balance
        self.transactions = []

    def add_transaction(self, transaction_type, amount, counterparty_id):
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        transaction = f"{timestamp} - {transaction_type}: ${amount} with User
{counterparty_id}"
        self.transactions.append(transaction)

class DigitalMoneyTransferSystem:
    def __init__(self):
        self.users = {}

    def create_user(self, user_id, initial_balance=0):
        if user_id not in self.users:
            self.users[user_id] = User(user_id, initial_balance)
            print(f"User {user_id} created with initial balance:
${initial_balance}")
        else:
            print(f"User {user_id} already exists.")

    def transfer_money(self, sender_id, receiver_id, amount):
        if sender_id not in self.users or receiver_id not in self.users:
            print("Invalid user IDs. Make sure both sender and receiver exist.")
            return

        sender = self.users[sender_id]
        receiver = self.users[receiver_id]

        if sender.balance < amount:
            print("Insufficient funds for the transfer.")
            return

        sender.balance -= amount
        receiver.balance += amount

        sender.add_transaction("Debit", amount, receiver_id)
```

```
        receiver.add_transaction("Credit", amount, sender_id)

        print(f"${amount} transferred from User {sender_id} to User
{receiver_id}")
        print(f"Updated balance for User {sender_id}: ${sender.balance}")
        print(f"Updated balance for User {receiver_id}: ${receiver.balance}")

    def list_user_balances(self):
        print("\nUser Balances:")
        for user_id, user in self.users.items():
            print(f"User {user_id}: ${user.balance}")

    def view_transaction_history(self, user_id):
        if user_id not in self.users:
            print("Invalid user ID. Make sure the user exists.")
            return

        user = self.users[user_id]
        print(f"\nTransaction History for User {user_id}:")
        for transaction in user.transactions:
            print(transaction)

# Example usage with extended functionality:
if __name__ == "__main__":
    dmts = DigitalMoneyTransferSystem()

    while True:
        print("\n1. Create User\n2. Transfer Money\n3. List User Balances\n4.
View Transaction History\n5. Exit")
        choice = input("Enter your choice (1/2/3/4/5): ")

        if choice == "1":
            user_id = input("Enter user ID: ")
            initial_balance = float(input("Enter initial balance: $"))
            dmts.create_user(user_id, initial_balance)
        elif choice == "2":
            sender_id = input("Enter sender's user ID: ")
            receiver_id = input("Enter receiver's user ID: ")
            amount = float(input("Enter the amount to transfer: $"))
            dmts.transfer_money(sender_id, receiver_id, amount)
        elif choice == "3":
            dmts.list_user_balances()
        elif choice == "4":
```

```
            user_id = input("Enter user ID to view transaction history: ")
            dmts.view_transaction_history(user_id)
        elif choice == "5":
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter 1, 2, 3, 4, or 5.")
```

Output :

```
python -u "/home/computer/Documents/BE/AIML11/aifb_exp1.py"
(base) computer@computer-ThinkCentre:~/Documents/BE/AIML11$

1. Create User
2. Transfer Money
3. List User Balances
4. View Transaction History
5. Exit
Enter your choice (1/2/3/4/5): 1
Enter user ID: prathamesh
Enter initial balance: $400
User prathamesh created with initial balance: $400.0

1. Create User
2. Transfer Money
3. List User Balances
4. View Transaction History
5. Exit
Enter your choice (1/2/3/4/5): 1
Enter user ID: sudham
Enter initial balance: $200
User sudham created with initial balance: $200.0

1. Create User
2. Transfer Money
3. List User Balances
4. View Transaction History
5. Exit
Enter your choice (1/2/3/4/5): 2
Enter sender's user ID: prathamesh
Enter receiver's user ID: sudham
Enter the amount to transfer: $100
$100.0 transferred from User prathamesh to User sudham
Updated balance for User prathamesh: $300.0
Updated balance for User sudham: $300.0
```

```
1. Create User
2. Transfer Money
3. List User Balances
4. View Transaction History
5. Exit
Enter your choice (1/2/3/4/5): 3

User Balances:
User prathamesh: $300.0
User sudham: $300.0

1. Create User
2. Transfer Money
3. List User Balances
4. View Transaction History
5. Exit
```

```
Enter your choice (1/2/3/4/5): 4
Enter user ID to view transaction history: sudham

Transaction History for User sudham:
2024-01-23 14:54:20 - Credit: $100.0 with User prathamesh

1. Create User
2. Transfer Money
3. List User Balances
4. View Transaction History
5. Exit
Enter your choice (1/2/3/4/5): 2
Enter sender's user ID: prathamesh
Enter receiver's user ID: sudham
Enter the amount to transfer: $500
Insufficient funds for the transfer.

1. Create User
2. Transfer Money
3. List User Balances
4. View Transaction History
5. Exit
Enter your choice (1/2/3/4/5): 5
Exiting the program. Goodbye!
(base) computer@computer-ThinkCentre:~/Documents/BE/AIML11$
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Code:

```
[3]  import pandas as pd
     import numpy as np
     import yfinance as yf

     # Portfolio and risk-free rate
     portfolio_tickers = ['AAPL', 'GOOGL', 'MSFT', 'AMZN']
     risk_free_rate = 0.01  # Adjust this based on the current risk-free rate

     # Download historical stock prices
     prices = yf.download(portfolio_tickers, start='2020-01-01', end='2023-01-01')

     # Display column names to see what's available
     print(prices.columns)

     # Now adjust the column name according to the available columns in the DataFrame
     # For example, if 'Close' is available, you can use it instead of 'Adj Close'

     # Calculate daily returns
     returns = prices['Close'].pct_change().dropna()

     # Equal-weighted portfolio returns
     portfolio_returns = returns.mean(axis=1)

     # Excess returns (portfolio returns - risk-free rate)
     excess_returns = portfolio_returns - risk_free_rate

     # Sharpe ratio calculation
     sharpe_ratio = excess_returns.mean() / excess_returns.std()
     print("Sharpe Ratio:", sharpe_ratio)
```

Output :

```
[*********************100%%**********************]  4 of 4 completedMultiIndex([('Adj Close',  'AAPL'),
            ('Adj Close',  'AMZN'),
            ('Adj Close', 'GOOGL'),
            ('Adj Close',  'MSFT'),
            (    'Close',  'AAPL'),
            (    'Close',  'AMZN'),
            (    'Close', 'GOOGL'),
            (    'Close',  'MSFT'),
            (     'High',  'AAPL'),
            (     'High',  'AMZN'),
            (     'High', 'GOOGL'),
            (     'High',  'MSFT'),
            (      'Low',  'AAPL'),
            (      'Low',  'AMZN'),
            (      'Low', 'GOOGL'),
            (      'Low',  'MSFT'),
            (     'Open',  'AAPL'),
            (     'Open',  'AMZN'),
            (     'Open', 'GOOGL'),
            (     'Open',  'MSFT'),
            (   'Volume',  'AAPL'),
            (   'Volume',  'AMZN'),
            (   'Volume', 'GOOGL'),
            (   'Volume',  'MSFT')],
           names=['Price', 'Ticker'])
Sharpe Ratio: -0.45833280107100083
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Output:

```
[1]  import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.cluster import KMeans
     import warnings
     warnings.filterwarnings('ignore')
```

```
[8]  df = pd.read_csv(r"mall_customers.csv")
```

```
[9]  df.head()
```

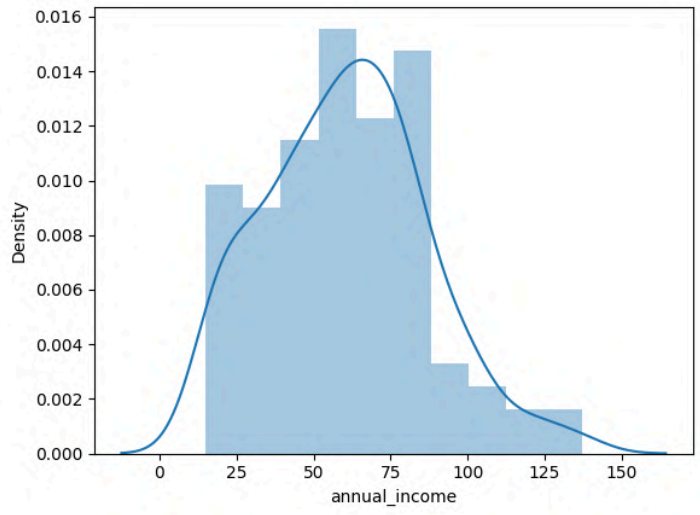
	customer_id	gender	age	annual_income	spending_score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Next steps: View recommended plots

```
[10] df.describe()
```

	customer_id	age	annual_income	spending_score
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

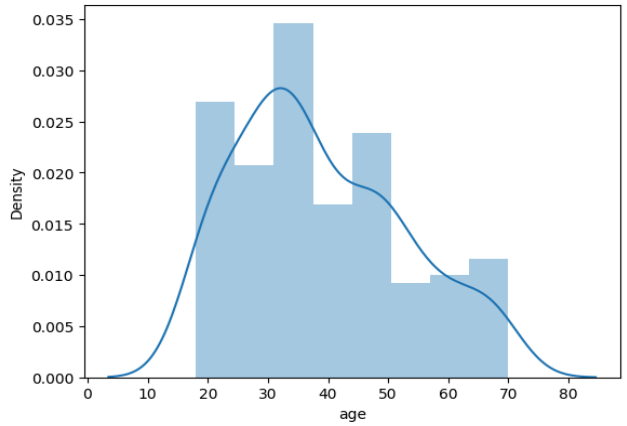
```
[12] sns.distplot(df['annual_income']);
```

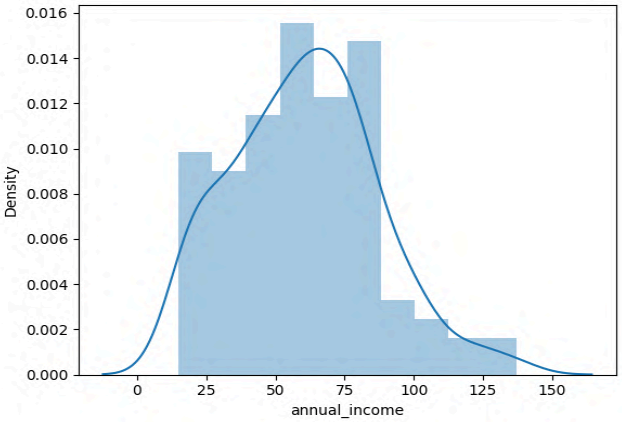


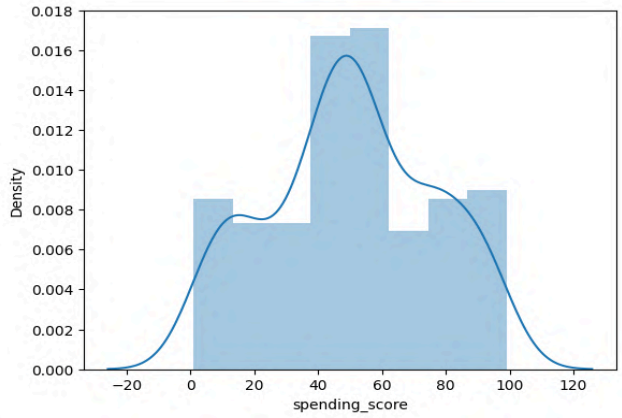
```
[13] df.columns
```

```
Index(['customer_id', 'gender', 'age', 'annual_income', 'spending_score'], dtype='object')
```

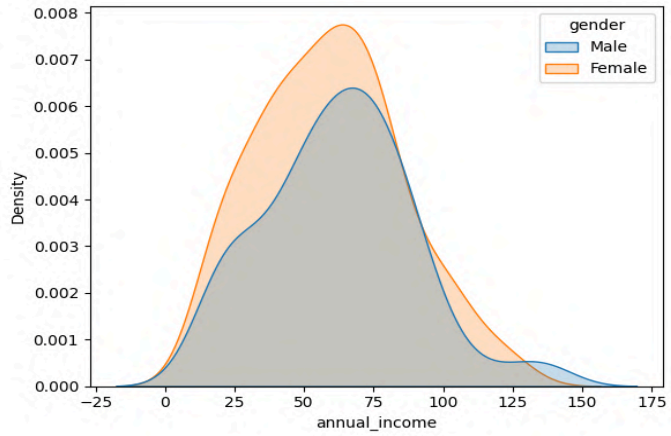
```
[15] columns = ['age', 'annual_income','spending_score']
     for i in columns:
       plt.figure()
       sns.distplot(df[i])
```








```
[20] sns.kdeplot(data=df, x='annual_income', shade=True, hue='gender')
```




```
[21] df['gender'].value_counts(normalize=True)
```

```
Female    0.56
Male      0.44
Name: gender, dtype: float64
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Code:

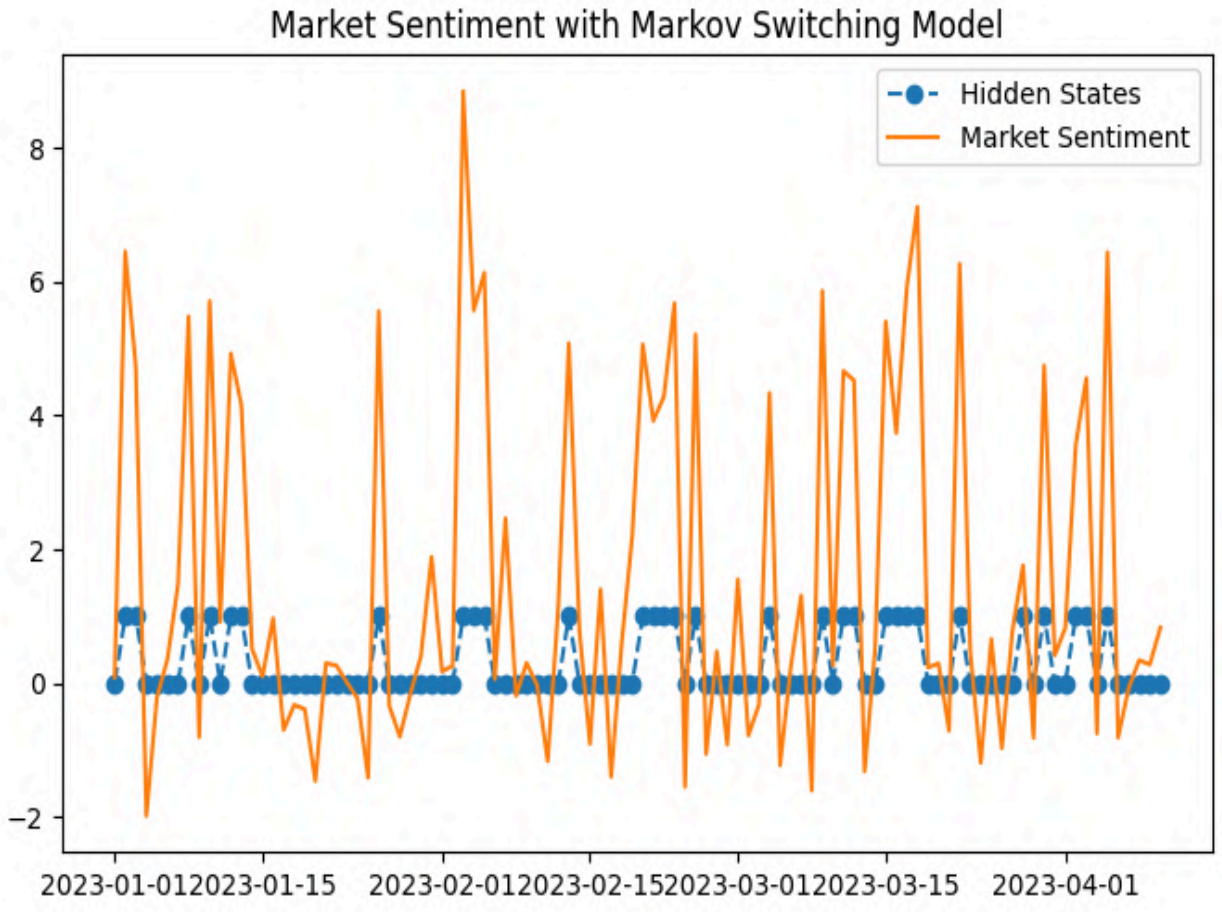
```
[2]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import statsmodels.api as sm

     # Generate synthetic data with two regimes
     np.random.seed(42)
     n_obs = 100
     regime_probs = [0.7, 0.3]
     hidden_states = np.random.choice([0, 1], size=n_obs, p=regime_probs)

     # Simulate market sentiment data
     sentiment_regime_0 = np.random.normal(0, 1, size=n_obs)
     sentiment_regime_1 = np.random.normal(5, 1, size=n_obs)
     market_sentiment = np.where(hidden_states == 0, sentiment_regime_0, sentiment_regime_1)
```

```
     # Create a DataFrame
     data = pd.DataFrame({'Market_Sentiment': market_sentiment}, index=pd.date_range('2023-01-01', periods=n_obs))
```

```
     # Fit a Markov Switching Model
     model = sm.tsa.MarkovRegression(data['Market_Sentiment'], k_regimes=2, trend='n')
     result = model.fit()

     # Print model summary
     print(result.summary())

     # Plot the hidden states and market sentiment
     fig, ax = plt.subplots(figsize=(8,5))
     ax.plot(data.index, hidden_states, linestyle='--', marker='o', label='Hidden States')
     ax.plot(data.index, data['Market_Sentiment'], label='Market Sentiment')
     ax.set_title('Market Sentiment with Markov Switching Model')
     ax.legend()
     plt.show()
```

Output:

```
                        Markov Switching Model Results
==============================================================================
Dep. Variable:       Market_Sentiment   No. Observations:                  100
Model:               MarkovRegression   Log Likelihood                -252.693
Date:                Sat, 23 Mar 2024   AIC                            511.386
Time:                        18:40:31   BIC                            519.201
Sample:                  01-01-2023     HQIC                           514.549
                       - 04-10-2023
Covariance Type:              approx
                          Non-switching parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         9.1704      1.297      7.071      0.000       6.629      11.712
                          Regime transition parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
p[0->0]        0.5000   1.63e+05   3.07e-06      1.000   -3.19e+05    3.19e+05
p[1->0]        0.5000   7.45e+04   6.71e-06      1.000   -1.46e+05    1.46e+05
==============================================================================

Warnings:
[1] Covariance matrix calculated using numerical (complex-step) differentiation.
```



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Output:

```
[1]  import os
     import math
     import numpy as np
     import pandas as pd
     from pylab import plt, mpl
     plt.style.use('seaborn')
     mpl.rcParams['savefig.dpi'] = 300
     mpl.rcParams['font.family'] = 'serif'
     pd.set_option('mode.chained_assignment', None)
     pd.set_option('display.float_format', '{:.4f}'.format)
     np.set_printoptions(suppress=True, precision=4)
     os.environ['PYTHONHASHSEED'] = '0'
```

```
<ipython-input-1-cb33373d5d47>:6: MatplotlibDeprecationWarning: The seaborn styles sh
  plt.style.use('seaborn')
```

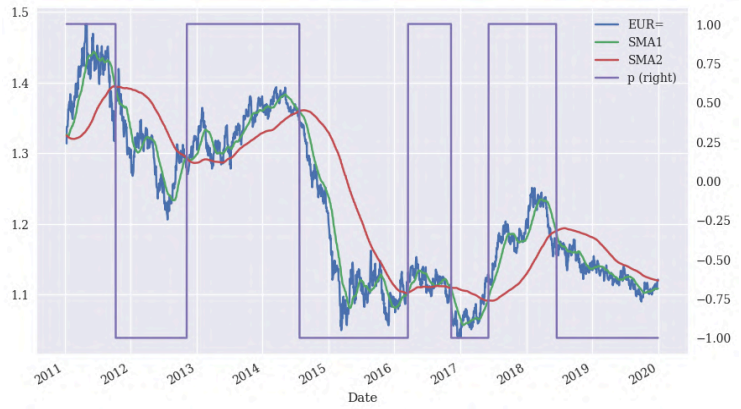
```
[2]  url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
```

```
[3]  symbol = 'EUR='
     data = pd.DataFrame(pd.read_csv(url, index_col=0,parse_dates=True).dropna()[symbol])
     data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2516 entries, 2010-01-04 to 2019-12-31
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   EUR=    2516 non-null   float64
dtypes: float64(1)
memory usage: 39.3 KB
```

```
[5]  In [3]: data['SMA1'] = data[symbol].rolling(42).mean()
     data['SMA2'] = data[symbol].rolling(258).mean()
     data.plot(figsize=(10, 6));
```


```
[6]  data.dropna(inplace=True)
     data['p'] = np.where(data['SMA1'] > data['SMA2'], 1, -1)
     data['p'] = data['p'].shift(1)
     data['p'] = data['p'].shift(1)
     data.dropna(inplace=True)
     data.plot(figsize=(10, 6), secondary_y='p');
```




```
[7]  data['r'] = np.log(data[symbol] / data[symbol].shift(1))
     data.dropna(inplace=True)
     data['s'] = data['p'] * data['r']
     data[['r', 's']].sum().apply(np.exp)
```

```
r    0.8534
s    1.4075
dtype: float64
```

```
[8]  data[['r', 's']].sum().apply(np.exp) - 1
```

```
r   -0.1466
s    0.4075
dtype: float64
```

```
[9]  data[['r', 's']].cumsum().apply(np.exp).plot(figsize=(10, 6));
```


```
[10] sum(data['p'].diff() != 0) + 2
```

```
10
```

```
[11] pc = 0.005
     data['s_'] = np.where(data['p'].diff() != 0,
     data['s'] - pc, data['s'])
     data['s_'].iloc[0] -= pc
     data['s_'].iloc[-1] -= pc
     data[['r', 's', 's_']][data['p'].diff() != 0]
```

Date	r	s	s_
2011-01-13	0.0165	0.0165	0.0065
2011-10-11	0.0001	-0.0001	-0.0051
2012-11-08	-0.0020	-0.0020	-0.0070
2014-07-25	-0.0025	0.0025	-0.0025
2016-03-17	0.0084	0.0084	0.0034
2016-11-11	-0.0036	0.0036	-0.0014
2017-06-06	0.0020	0.0020	-0.0030
2018-06-18	0.0013	-0.0013	-0.0063

```
[12] data[['r', 's', 's_']].sum().apply(np.exp)
```

```
r    0.8534
s    1.4075
s_   1.3388
dtype: float64
```

```
[13] data[['r', 's', 's_']].sum().apply(np.exp) - 1
```

```
r    -0.1466
s     0.4075
s_    0.3388
dtype: float64
```

```
[14] data[['r', 's', 's_']].cumsum().apply(np.exp).plot(figsize=(10, 6))
```

```
<Axes: xlabel='Date'>
```


```
[15] data[['r', 's', 's_']].std()
```

```
r    0.0054
s    0.0054
s_   0.0054
dtype: float64
```

```
[16] data[['r', 's', 's_']].std() * math.sqrt(252)
```

```
r    0.0853
s    0.0852
s_   0.0851
dtype: float64
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Code & Output:

```
[2] import numpy as np
    import pandas as pd
    import yfinance as yf # You may need to install this package
```

```
[3] # Download historical data
    ticker = "AAPL"
    start_date = "2020-01-01"
    end_date = "2022-01-01"
    data = yf.download(ticker, start=start_date, end=end_date)
```

```
[**********100%%**********]  1 of 1 completed
```

```
[4] # Define parameters
    short_window = 40
    long_window = 100
    risk_per_trade = 0.02 # Risk 2% of equity per trade
```

```
[6] # Calculate moving averages
    data['Short_MA'] = data['Close'].rolling(window=short_window, min_periods=1).mean()
    data['Long_MA'] = data['Close'].rolling(window=long_window, min_periods=1).mean()
```

```
[7] # Create signals
    data['Signal'] = 0
    data['Signal'][short_window:] = np.where(data['Short_MA'][short_window:] > data['Long_MA'][short_window:], 1, 0)
```

```
<ipython-input-7-f3bb0f2f13ed>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['Signal'][short_window:] = np.where(data['Short_MA'][short_window:] > data['Long_MA'][short_window:], 1, 0)
```

```
[8] # Calculate daily returns
    data['Daily_Returns'] = data['Close'].pct_change()
```

```
[9] # Implement basic risk management
    data['Position_Size'] = 0
    data['Position_Size'][data['Signal'] == 1] = risk_per_trade / (data['Close'][data['Signal'] == 1] - data['Short_MA'][data['Signal'] == 1])
```

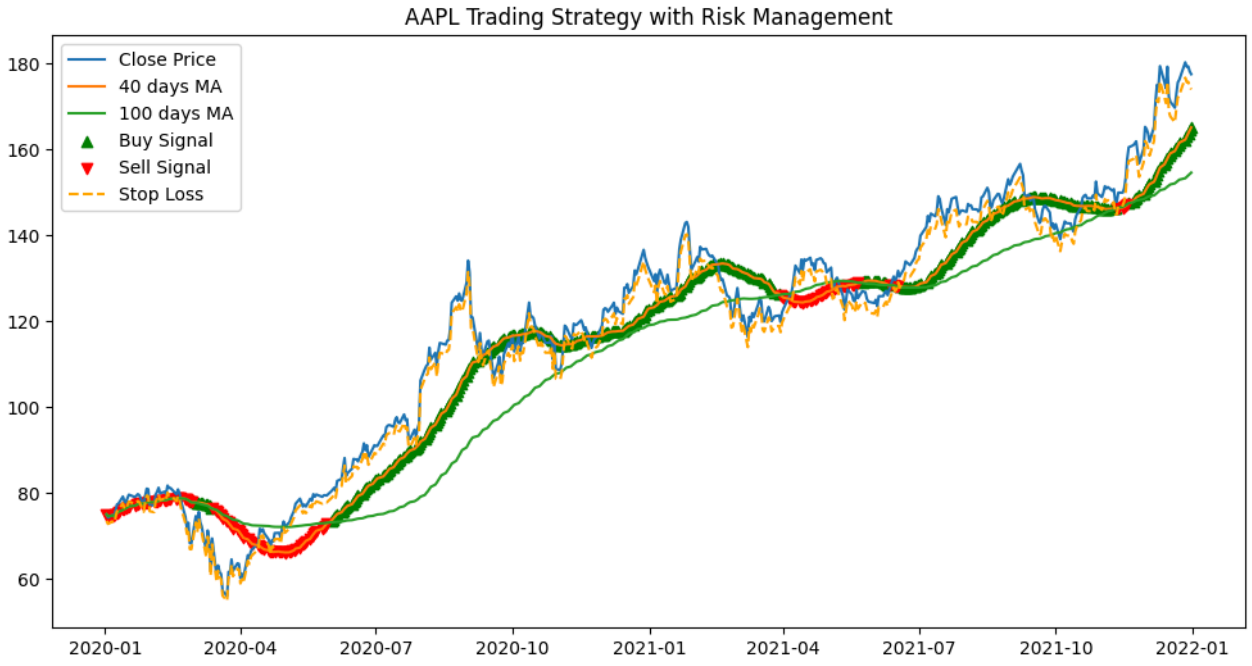
```
<ipython-input-9-8f7e916bd17b>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['Position_Size'][data['Signal'] == 1] = risk_per_trade / (data['Close'][data['Signal'] == 1] - data['Short_MA'][data['Signal'] == 1])
```

```
[10] # Calculate stop-loss price
     data['Stop_Loss'] = data['Close'] * (1 - 0.02) # 2% stop loss
```

```
[11] # Calculate equity curve
     data['Portfolio_Value'] = data['Position_Size'] * data['Daily_Returns'] + 1
     data['Equity_Curve'] = data['Portfolio_Value'].cumprod()
```

```
[12] # Plot the results
     import matplotlib.pyplot as plt
```

```
[13] plt.figure(figsize=(12, 6))
     plt.plot(data['Close'], label='Close Price')
     plt.plot(data['Short_MA'], label=f'{short_window} days MA')
     plt.plot(data['Long_MA'], label=f'{long_window} days MA')
     plt.scatter(data[data['Signal'] == 1].index, data['Short_MA'][data['Signal'] == 1], marker='^', color='green', label='Buy Signal')
     plt.scatter(data[data['Signal'] == 0].index, data['Short_MA'][data['Signal'] == 0], marker='v', color='red', label='Sell Signal')
     plt.plot(data['Stop_Loss'], label='Stop Loss', linestyle='--', color='orange')
     plt.title(f'{ticker} Trading Strategy with Risk Management')
     plt.legend()
     plt.show()
```



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Code & Output:

In [6]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import chardet

# Detect the encoding of the file
with open('fraud-1.csv', 'rb') as f:
    result = chardet.detect(f.read())

# Use the detected encoding to read the CSV file
df = pd.read_csv('fraud-1.csv', encoding=result['encoding'])

# Print column names
print("Column Names:")
print(df.columns)

# Assuming the target variable is 'Fraud' (1 for fraud, 0 for non-f
# Make sure 'Fraud' is present in the column names
if 'Fraud' in df.columns:
    X = df.drop('Fraud', axis=1)
    y = df['Fraud']

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_

    # Create a Random Forest Classifier model
    model = RandomForestClassifier(n_estimators=100, random_state=4

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(X_test)

    # Print actual vs predicted values
    result_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred
    print("\nActual vs Predicted:")
    print(result_df)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)

    # Display results
    print(f"\nAccuracy: {accuracy}")
    print("Confusion Matrix:")
    print(conf_matrix)

else:
    print("Column 'Fraud' not found in the DataFrame. Please check
```

```
Column Names:
Index(['Fraud', '2013/14', '2014/15', '2015/16', '2016/17', '2017/
18',
       '2018/19'],
      dtype='object')

Actual vs Predicted:
                                                 Actual  \
0  Number of occasions powers under the Preventio...
1  Total number of employees undertaking investig...

                                              Predicted
0  Total full time equivalent of professionally a...
1  Total full time equivalent of employees undert...

Accuracy: 0.0
Confusion Matrix:
[[0 0 1 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 1 0 0]]
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Code & Output:

```
[2]  import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     # Create a synthetic dataset
     data = {
         'TransactionID': list(range(1, 101)),
         # Adjusted the length
         'Amount': [100, 150, 200, 500, 100, 120, 180, 450, 90, 200] * 10, # Adjusted the length
         'IsFraud': [0, 0, 0, 1, 0, 0, 0, 1, 0, 0] * 10, # Adjusted the length
         'Location': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'C', 'B', 'A'] * 10,
         # Add more relevant features based on your actual data
     }
     df = pd.DataFrame(data)

     # Visualize transaction amounts for fraud and non-fraud transactions
     plt.figure(figsize=(10, 6))
     colors = ['green' if fraud == 0 else 'red' for fraud in df['IsFraud']]
     plt.scatter(df['TransactionID'], df['Amount'], c=colors)
     plt.title('Transaction Amounts - Fraud vs. Non-Fraud')
     plt.xlabel('Transaction ID')
     plt.ylabel('Amount')
     plt.show()

     # Visualize fraud distribution by location
     plt.figure(figsize=(8, 5))
     # Corrected the color palette
     sns.countplot(x='Location', hue='IsFraud', data=df, palette={0: 'green', 1: 'red'})
     plt.title('Fraud Distribution by Location')
     plt.xlabel('Location')
     plt.ylabel('Count')
     plt.show()
```

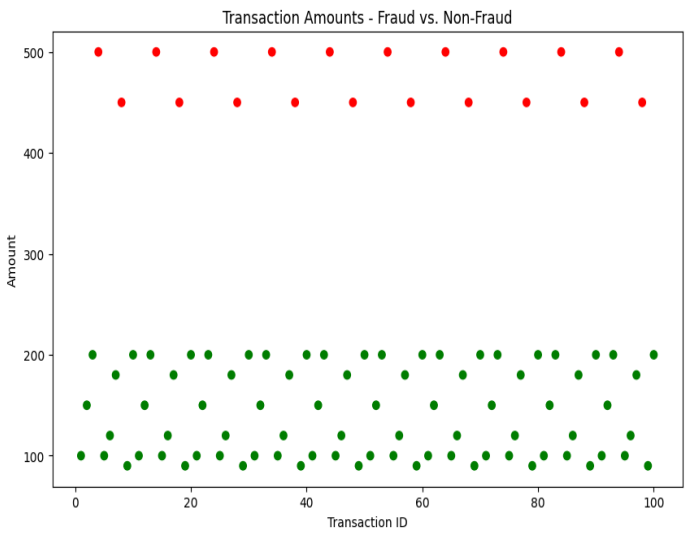


Fig. 1

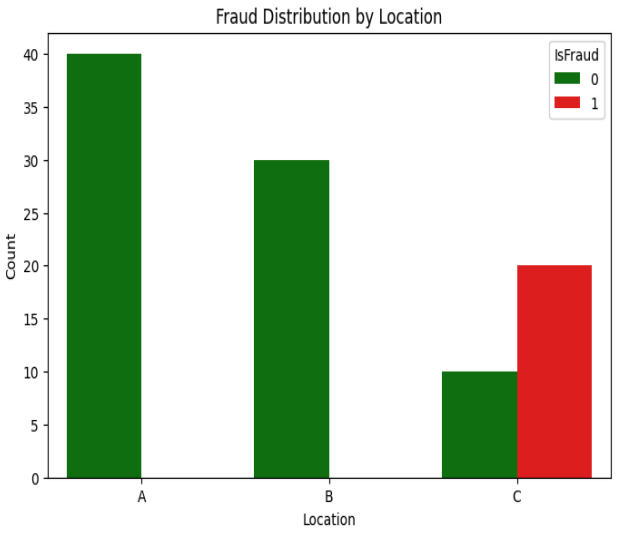


Fig. 2

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

EXPERIMENT NO. 09

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Define Strategy Logic
def simple_moving_average_strategy(data, short_window=20, long_window=50):
    """
    Simple Moving Average (SMA) Crossover Strategy
    Buy when short-term SMA crosses above long-term SMA, sell when short-term SMA crosses
below long-term SMA.
    """
    signals = pd.DataFrame(index=data.index)
    signals['Signal'] = 0

    # Compute short-term SMA
    signals['Short_SMA'] = data['Close'].rolling(window=short_window, min_periods=1).mean()

    # Compute long-term SMA
    signals['Long_SMA'] = data['Close'].rolling(window=long_window, min_periods=1).mean()

    # Generate buy signals
    signals['Signal'][short_window:] = np.where(
        signals['Short_SMA'][short_window:] > signals['Long_SMA'][short_window:], 1, 0
    )

    # Generate sell signals
    signals['Signal'][short_window:] = np.where(
        signals['Short_SMA'][short_window:] < signals['Long_SMA'][short_window:], -1,
signals['Signal'][short_window:]
    )

    return signals

# Step 2: Backtest Strategy
def backtest_strategy(strategy, data):
    # Apply the strategy to the data
    signals = strategy(data)

    # Compute returns based on signals
    returns = data['Close'].pct_change() * signals['Signal'].shift(1)

    # Fill NaN values in returns with 0
    returns.fillna(0, inplace=True)

    # Combine returns and signals into a portfolio dataframe
    portfolio = pd.DataFrame({
        'Asset_Position': signals['Signal'],
```

```
        'Cash': 1 - signals['Signal'],
        'Total': 1,
        'Returns': returns
    }, index=data.index)  # Add index parameter to DataFrame constructor

    return portfolio

# Step 3: Performance Evaluation
def evaluate_strategy(portfolio):
    # Calculate performance metrics
    cumulative_returns = (1 + portfolio['Returns']).cumprod()

    print(f"Cumulative Returns:\n{cumulative_returns}")

    # Check if there are any NaN values in returns
    if portfolio['Returns'].isnull().values.any():
        print("WARNING: NaN values found in returns data!")
        print(portfolio[portfolio['Returns'].isnull()])  # Debug print to identify rows with
NaN returns

    # Check if there are any trades made by the strategy
    if np.isnan(portfolio['Returns']).all():
        sharpe_ratio = 0.0
        max_drawdown = 0.0
    else:
        # Filter out -inf values
        valid_returns = portfolio['Returns'][~np.isinf(portfolio['Returns'])]
        if valid_returns.empty:
            sharpe_ratio = 0.0
            max_drawdown = 0.0
        else:
            sharpe_ratio = np.sqrt(252) * (valid_returns.mean() / valid_returns.std())
            max_drawdown = np.max(np.maximum.accumulate(cumulative_returns) -
cumulative_returns)

    print(f"Sharpe Ratio: {sharpe_ratio:.2f}")
    print(f"Maximum Drawdown: {max_drawdown:.2f}")

    return cumulative_returns, sharpe_ratio, max_drawdown

# Step 4: Example Usage
# Load historical price data (e.g., from a CSV file or using an API)
# For simplicity, let's generate random price data for this example
np.random.seed(42)
dates = pd.date_range(start='2021-01-01', end='2021-12-31')
prices = 100 + np.cumsum(np.random.normal(0, 1, len(dates)))
data = pd.DataFrame({'Close': prices}, index=dates)

# Apply the trading strategy
signals = simple_moving_average_strategy(data)
```

```
# Backtest the strategy
portfolio = backtest_strategy(simple_moving_average_strategy, data)

# Evaluate strategy performance
cumulative_returns, sharpe_ratio, max_drawdown = evaluate_strategy(portfolio)

# Plot results
plt.figure(figsize=(12, 6))
plt.plot(cumulative_returns.index, cumulative_returns, label='Strategy Returns')
plt.title('Cumulative Returns')
plt.xlabel('Date')
plt.ylabel('Returns')
plt.legend()
plt.grid(True)
plt.show()
```

Output :

```
Cumulative Returns:
2021-01-01    1.000000
2021-01-02    1.000000
2021-01-03    1.000000
2021-01-04    1.000000
2021-01-05    1.000000
                ...
2021-12-27    0.941695
2021-12-28    0.927420
2021-12-29    0.928403
2021-12-30    0.924766
2021-12-31    0.918566
Freq: D, Name: Returns, Length: 365, dtype: float64
Sharpe Ratio: -0.30
Maximum Drawdown: 0.26
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

EXPERIMENT NO. 10

Theory:

❖ **Defining a Trading Algorithm**

A trading algorithm, often referred to as an algo or automated trading system , on platforms like uTrade Algos, is a set of predefined rules and instructions designed to execute
financial transactions automatically.

- Leveraging mathematical models and data analysis, these algorithms, in algo trading, aim to capitalize on market opportunities by swiftly processing large volumes of information and executing trades with precision.
- Trading algorithms can be designed to perform various functions, such as trend following, statistical arbitrage, or market making.
- Their primary goal is to remove emotional bias, ensure consistent execution, and enhance efficiency in financial markets, catering to both institutional and individual traders seeking a systematic and data-driven approach to trading.

❖ **Need For Evaluating Trading Algorithms**

In automated algorithmic trading , on platforms like uTrade Algos, evaluating trading algorithms is crucial to gain insights into their performance, manage risks, and optimize strategies for sustained success.

- Performance Insights: Gain a comprehensive understanding of the algorithm's ability to generate consistent returns.
- Risk-Adjusted Returns: Evaluate risk-adjusted performance through metrics like the Sharpe ratio, ensuring favorable returns relative to the level of risk involved.
- Accuracy Assessment: Analyze winning percentages to gauge the algorithm's accuracy in making successful trades, providing insights for informed decision-making.
- Risk Management: Monitor metrics such as maximum drawdown to effectively manage risks, ensuring the algorithm's stability across diverse market conditions.
- Adaptability and Optimisation: Identify areas for improvement and optimisation, allowing traders to fine-tune parameters and enhance the algorithm's adaptability in dynamic market environments.

❖ **Key Metrics to Evaluate Performance of Trading Algorithms**

➢ **Financial Viability**

While algo trading, one should:

- Scrutinize the trading algorithm's effectiveness in generating favorable outcomes over a designated period.
- Examine the algorithm's ROI to understand the quantitative measure of returns relative to the initial investment.
- Assess the algorithm's consistency in delivering positive financial outcomes without guaranteed profits.
- Gain valuable insights into how the algorithm contributes to achieving financial objectives without assurance of profit.

➢ **Sharpe Ratio**

Assess the risk-adjusted performance using the Sharpe ratio.

- It calculates the excess return per unit of risk, factoring in the risk-free rate and standard deviation.
- A higher Sharpe ratio signifies superior risk-adjusted returns, indicating efficient gains relative to risk exposure.
- It offers insights into the algorithm's performance, aiding in decisions about risk and returns.
- It quantifies the algorithm's efficiency in converting risk into returns, facilitating comparisons for optimal strategy selection.

➢ **Winning Percentage**

- The winning percentage serves as a crucial metric, directly gauging the algorithm's accuracy in generating profits without guaranteeing success. A higher winning percentage signifies a more effective algorithm, consistently yielding positive outcomes and showcasing reliability over time. This metric offers valuable insights for traders in decision- making, providing a measurable benchmark to assess the algorithm's success in executing profitable trades, albeit without an assured guarantee of profitability.

➢ **Maximum Drawdown**

- Measure the maximum drawdown, representing the largest peak-to-trough decline in your algorithm's equity curve.
- A lower drawdown indicates better risk management and stability.
- It serves as a tool to measure stability, with a lower drawdown indicating increased resilience to market fluctuations.
- It is used to assess investor confidence, instilling trust by limiting losses and enhancing overall attractiveness to investors.
- It helps find the right balance between risk and return, assessing the algorithm's capacity to navigate challenging market conditions while preserving capital.

➢ **Volatility Metrics**

- Analyze volatility measures such as standard deviation to understand the algorithm's risk exposure. Standard deviation assesses the extent of price fluctuations, showcasing the algorithm's sensitivity to market volatility.
- By analyzing volatility, traders gain insights into the algorithm's ability to navigate dynamic market conditions.
- Balancing returns with volatility becomes crucial, as it ensures a more stable and predictable trading strategy.
- Careful consideration of volatility allows traders to optimize risk-return profiles, aligning the algorithm with their risk tolerance and overall trading objectives.
- This nuanced approach enables the development of strategies that thrive in varying market environments while maintaining stability and predictability.

Code:

In [14]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
```

In [15]:
```
# Step 1: Define Performance Metrics
def calculate_sharpe_ratio(returns):
    sharpe_ratio = np.sqrt(252) * (np.mean(returns) / np.std(returns))
    return sharpe_ratio

def calculate_max_drawdown(returns):
    cum_returns = np.cumprod(1 + returns)
    max_drawdown = np.max(np.maximum.accumulate(cum_returns) - cum_returns)
    return max_drawdown
```

In [16]:
```
# Step 2: Backtesting
def backtest_strategy(data, signal):
    returns = data['Adj Close'].pct_change().shift(-1) * signal
    return returns
```

In [17]:
```
# Step 3: Transaction Costs and Slippage
transaction_cost = 0.001  # 0.1% transaction cost
```

In [18]:
```
# Step 4: Risk Management (Position Sizing)
def position_size(account_value, risk_per_trade, entry_price, stop_loss_price):
    position_size = (account_value * risk_per_trade) / (entry_price - stop_loss_price)
    return position_size
```

In [19]:
```
# Step 5: Robustness and Sensitivity Analysis
def parameter_sensitivity_analysis(data, parameters):
    # Example: Test different moving average lengths
    results = {}
    for param in parameters:
        signal = np.where(data['Adj Close'] > data['Adj Close'].rolling(param).mean(), 1, -1)
        returns = backtest_strategy(data, signal)
        results[param] = calculate_sharpe_ratio(returns)
    return results
```

In [20]:
```
# Step 6: Machine Learning Techniques (Example: Random Forest Classifier)
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

def train_ml_model(data):
    X = data[['Open', 'High', 'Low', 'Volume']]
    y = np.where(data['Adj Close'].shift(-1) > data['Adj Close'], 1, -1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    accuracy = model.score(X_test, y_test)
    return model, accuracy
```

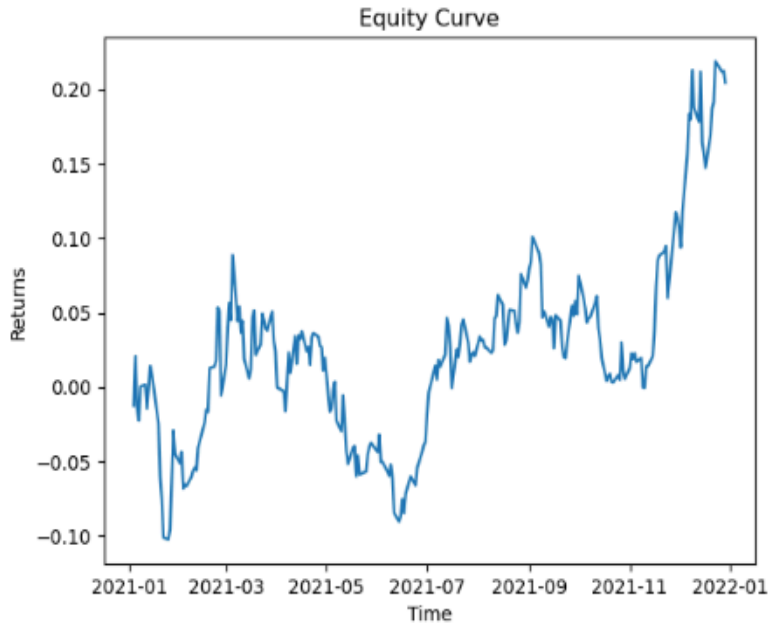
In [21]:
```
# Step 7: Live Trading and Paper Trading (Example: Using Yahoo Finance)
def get_live_data(symbol, start_date, end_date):
    data = yf.download(symbol, start=start_date, end=end_date)
    return data
```

In [22]:
```
# Step 8: Continuous Monitoring and Optimization
# Example: Continuously monitor and adjust moving average crossover strategy based on recent performance

# Step 9: Compliance and Risk Assessment
# Implement risk management rules, regulatory compliance checks, etc.

# Step 10: Documentation and Reporting
# Document the entire process, including strategy development, backtesting results, and performance evaluation.
```

In [23]:
```
# Example usage
symbol = 'AAPL'
start_date = '2021-01-01'
end_date = '2021-12-31'

data = get_live_data(symbol, start_date, end_date)
signal = np.where(data['Adj Close'] > data['Adj Close'].rolling(50).mean(), 1, -1)
returns = backtest_strategy(data, signal)
sharpe_ratio = calculate_sharpe_ratio(returns)
max_drawdown = calculate_max_drawdown(returns)

print("Sharpe Ratio:", sharpe_ratio)
print("Max Drawdown:", max_drawdown)

plt.plot(np.cumprod(1 + returns) - 1)
plt.title("Equity Curve")
plt.xlabel("Time")
plt.ylabel("Returns")
plt.show()
```

Output :

```
[**********100%%**********]  1 of 1 completed

Sharpe Ratio: 0.8740390462156878
Max Drawdown: 0.17910944172155385
```



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VIII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**AI FOR FINANCE & BANKING APPLICATION LAB**
COURSE CODE	**CSDOL8011**
PRACTICAL NO.	
DOP	
DOS	

Code & Output -

```
[1] # Import necessary libraries
    import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
[2] # Generate a random dataset for demonstration purposes
    np.random.seed(42)
    data_size = 10000
```

```
[3] # Features
    transaction_amounts = np.random.uniform(1, 1000, data_size)
    transaction_dates = np.random.choice(pd.date_range('2022-01-01', '2024-01-01', freq='H'), data_size)
    is_fraudulent = np.random.choice([0, 1], data_size, p=[0.95, 0.05])
```

```
[4] # Create DataFrame
    df = pd.DataFrame({'TransactionAmount': transaction_amounts,'TransactionDate': transaction_dates,
    'IsFraudulent': is_fraudulent})
```

```
[5] df
```

	TransactionAmount	TransactionDate	IsFraudulent
0	375.165579	2022-02-05 23:00:00	0
1	950.763592	2023-09-13 03:00:00	0
2	732.261948	2023-05-22 01:00:00	0
3	599.059826	2022-05-08 07:00:00	0
4	156.862622	2023-05-23 18:00:00	0
...	...	...	...
9995	857.798332	2023-12-11 19:00:00	0
9996	897.611326	2022-02-13 18:00:00	0
9997	946.761207	2023-07-11 13:00:00	0
9998	398.090504	2022-07-17 14:00:00	0
9999	217.923264	2023-07-12 01:00:00	0

10000 rows × 3 columns

```
[6] # Feature engineering: Extract hour and day of week from transaction date
    df['HourOfDay'] = df['TransactionDate'].dt.hour
    df['DayOfWeek'] = df['TransactionDate'].dt.dayofweek
    df
```

	TransactionAmount	TransactionDate	IsFraudulent	HourOfDay	DayOfWeek
0	375.165579	2022-02-05 23:00:00	0	23	5
1	950.763592	2023-09-13 03:00:00	0	3	2

2	732.261948	2023-05-22 01:00:00	0	1	0
3	599.059826	2022-05-08 07:00:00	0	7	6
4	156.862622	2023-05-23 18:00:00	0	18	1
...	...	...	...	...	...
9995	857.798332	2023-12-11 19:00:00	0	19	0
9996	897.611326	2022-02-13 18:00:00	0	18	6
9997	946.761207	2023-07-11 13:00:00	0	13	1
9998	398.090504	2022-07-17 14:00:00	0	14	6
9999	217.923264	2023-07-12 01:00:00	0	1	2

10000 rows × 5 columns

Next steps: View recommended plots

0s
```
[7] # Drop the original date column
    df = df.drop('TransactionDate', axis=1)
```

0s
```
[8] # Split the data into training and testing sets
    X = df.drop('IsFraudulent', axis=1)
    y = df['IsFraudulent']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2s
```
[9] # Initialize and train a Random Forest classifier
    clf = RandomForestClassifier(n_estimators=100, random_state=42)
    clf.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

0s
```
[10] y_pred = clf.predict(X_test)
     prediction = pd.DataFrame({"actual":y_test,"prediction":y_pred})
     prediction
```

	actual	prediction
6252	0	0
4684	0	0
1731	0	0
4742	0	0
4521	0	0
...	...	...
6412	0	0
8285	0	0

7853	0	0
1095	0	0
6929	0	0

2000 rows × 2 columns

Next steps: View recommended plots

```
[11] # Evaluate the model
     accuracy = accuracy_score(y_test, y_pred)
     conf_matrix = confusion_matrix(y_test, y_pred)
     class_report = classification_report(y_test, y_pred)

     # Print the results
     print(f'Accuracy: {accuracy:.2f}')
     print('\nConfusion Matrix:')
     print(conf_matrix)
     print('\nClassification Report:')
     print(class_report)
```

```
Accuracy: 0.94

Confusion Matrix:
[[1879   33]
 [  85    3]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1912
           1       0.08      0.03      0.05        88

    accuracy                           0.94      2000
   macro avg       0.52      0.51      0.51      2000
weighted avg       0.92      0.94      0.93      2000
```

Name : Singh Sudham Dharmendra.

Branch :- (SE(AIML)

Roll no :- AIML

Subject :- Artificial Intelligence in finance & Banking System.

Topic :- Assignment No: (1)

Date of submission :-

Signature :-

Q.1] Explain benefits of Blockchain Technology in the banking sector.

→ ① Costs reduced: One of the benefits of blockchain for banks is reduced costs. Bank have recently learned that blockchain can allow them to reduce infrastructure costs upto $20 billion by year 2022. Bank can reduce cost b/w bank to bank transaction (smart contract).

② Faster transaction: Any transaction can be done with in a matter of seconds and slightly faster than other traditional methods.

③ improved security: Shared ledgers can help banks better secure transaction information. Two savity keys exist for each transaction. Data is unchangeable after verified.

④ improved data quality: modern blockchain can store any type of data and allow it to be accessed following predefined rules & regulation. Smart contracts automatically verifies & enforces contract.

⑤ Digital Currencies: Banks can benefit from blockchain within the use of digital currencies. They are now able to accept digital currency to complete a variety of transaction.

⑥ Accountability: Reducing fraud & misuse of company assests.

⑦ Reduced Error Handling: Allowing more easily reconcile transaction Trace transaction more quickly & find error in timeline manner.

Q.2] Write a short note on GIFT City.

→ GIFT city, which stands for Gujarat International Finance Tee-City is a planned business districts in the Indian State of Gujarat. The project is intended to promise international financial services and create a business-friendly environment.

It is designed to be global financial and IT services hub & is located b/w Ahemdabad & Gandhinagar, Capital city of Gujarat Features and aspects of GIFT City.

① Special Economic zone (SEZ): offering various incentives to attract business, particularly those in financial & IT sectors.

② Financial Services Hub: providing a platforms for various activities.
③ Infrastructure & connectivity: designed with modern infrastructure, including high-quality office space & connectivity through road.
④ International exchange: Hoses the IFSC, includes international exchange for trading various financial instruments & currencies.
⑤ Regulatory framework: within Indian legal system [regulation]
⑥ Smart city features: enhance efficiency & sustainability.
⑦ Global Companies & institution
⑧ Government Support.

Q.3] Explain digital Money transfer Mechanism:

→ Involves electronic movement of funds from one party to another.

① Online Banking: Bank transfer - within same bank (money transfer) and wire transfer - Electronic funds transfer [bank to bank]
② Mobile banking Apps: Peer-to-peer (P2P) payments - users can transfer money directly to friends or family through mobile apps, mobile wallets - example - Googlepay, Apple pay.
③ Digital wallets: e-wallets - platforms like pay-pal, venmo cryptocurrency wallets: cbyptocurrencies like bitcoin → digital wallets
④ Peer-to-Peer (P2P) platforms: P2P leading platfoam - source platform enable users to lend & borrow money directly from one another
⑤ Cryptocurrencies transfer: BCT - ensuring secure & transparent transaction
⑥ Cross-border payment services: Services like western union.
⑦ Digital banks: Neo banks - fully digital Bank.
⑧ Payment Gateways: Onlin Merchens - platfoam like Stripe, square
⑨ QR code payments: Scaning QR codes - Google pay.
⑩ Central Bank Digital Currencies (CBDC): Source cantry are exploring & implementing their own currency issied by central bank.

Q.4] Explain application of AI in financial & Banking Sector.

→ ① Cyber security & fraud detection: AI-powered Cyber-Security System analyze N/w traffic and user behaviour to detect & prevent cyber-threats & identifying fraudlent.

② Customer Experience and chatbots: AI-driven chatbots provide personalized assistance, answering customer queries.

③ Risk Management: assess various risk factors, including market volatility & ~~and~~ credit worthiness & minimize losses

④ Loan & Credit decision process: Analyze diverse data surce to evaluate loan application, accurately accessing borrower's credit worthiness & streamlining loan approval process

⑤ Tracking Market Trends: Analyze large volume of market data in real-time to identify trends, pattern & opportunities

⑥ Data Collection & Analysis: AI system collect & analyzes vast amount of data · structured & unstructured from diverse sources, providing insights into customer behaviour.

⑦ Predictive Analysis in regulatory compliance: involves leveraging AI & ML to anticipate potential compliance risks, identify process pattern of non-compliance & predict regulatory changes.

Q.5] Explain digitization in the Financial & banking sectors.

→ Reshaping traditional processes, enhancing customer experience & improving overall efficiency.

① Digital Banking: online-banking - customer can access their accounts, transfer funds & transaction through secure online portals. mobile banking - accessible via mobile device including smartphones

② Cloud-based core Banking: Core processing Systems - provide centralized system/platform for managing essential functions of banking such as deposits, loan & transaction.

③ Customer Relationship Management (CRM): Personalized service. 'CRM System help in understanding customer preferences, enabling banks to offer personalized services & target marketing

④ Data Analytics & business intelligence: customer insights - utilizing analytics to gain valuable insights into customers - behaviours & Risks management predictive analysis helpful in this.

⑤ Cybersecurity solution: Cloud based security - utilizing cloud-based security services to product sensitive financial data from cyber crime.

⑥ API (application programming interface): Open banking APIs facilitate secure data sharing between financial institution, enabling the development of involves financial products & services.

⑦ Digital wallets & Payments ⑧ Blockchain & ledger.

⑨ AI-powered chatbots. ⑩ Digital lending platform.

Q.6] Explain the skewness & kurtosis with formulas.

→ Skewness: Skewness measures the assymetry of the distribution of data around its mean. Skewness of zero indicates a symmetric distribution, while +ve & -ve skewness indicates skew to right or left respectively

The sample skewness is computed from the dataset. It can be calculated as,

$$\text{Skew}(R) = \frac{1}{N} \sum_{i=1}^{N} \left\{ \frac{R_i - \bar{R}}{s} \right\}^3$$

In R, the skewness (x) fn from moments library will use moment method above to compute the sample skewness,

where n represents → no. of observations in sample.

R_i & $\bar{R}$ → each observation & sample mean.

s → sample standard deviation.

Kurtosis :- Measures the degree of peakedness or flatness after attribution compared to normal distribution positive kurtosis indicates a sharper peak (leptokurtic), while negative kurtosis indicate flatter peak (Playkurtic) compared to normal distribution. The sample kurtosis is also computed from the dataset. It can be calculated as, $kurt(R) = \frac{1}{N} \sum_{i=1}^{N} \left\{ \frac{(R_i - \bar{R})}{s} \right\}^4$

In R, the kurtosis (x) function from the moments library will use the moments method above to compute the sample kurtosis.

Q.7] Explain sample covariance & correlation with formulas?

→ Sample covariance : Measures the degree to which two variables change together. A positive covariance indicates that the variable tend to increase or decrease together, while negative covariance indicates that one variable tends to increase as other decreases defined as, $\Sigma = Cov(R_j, R_x) = E\{(R_j - \bar{R_j})(R_x - \bar{R_x})\}$

where, E → Expected value and R → random variable

also; $\bar{R_j} = \frac{1}{N} \sum_{i=1}^{N} R_{ij}$ & $\bar{R}_x = \frac{1}{N} \sum_{i=1}^{N} R_{i,x}$

Correlation : sample correlation is a standardized measure of strength and direction of linear relationship b/w two variables. It is obtained by dividing sample covariance by the product of the sample standard deviations of the two variables.

defined as, $cor(R_j, R_x) = diag(\Sigma)^{1/2} \Sigma\, diag(\Sigma)^{1/2}$

Correlation coefficient ranges b/w -1 & 1

Q.8] Explain Sample mean, Standard deviation & Variance with example.

→ Sample mean :- often denoted by $\bar{R}$, represents, the average value of a dataset calculated by summing up all the values in the dataset & then dividing no. of observation.

Eg:- Consider dataset for exam score : {85, 90, 88, 92, 86}
the sample media $\bar{R}$ can be calculated as $\bar{R} = \frac{1}{N} \sum_{i=1}^{N} R_i$

$\therefore \bar{R} = \frac{85+90+88+92+86}{5} = \frac{441}{5} = 88.2$ ⇐ Sample mean of dataset.

Standard deviation : often denoted by s, measure ~~dataset~~ spread or dispersion of values in dataset around the mean. quantifies how much the values in dataset ~~deviate~~ from the average

$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (R_i - \bar{R})^2}$. } using same dataset

example, Calculate diff. blw each score. and mean

{-3.2, 1.8, -0.2, 3.8, -2.2} Square of each diff.

{10.24, 3.24, 0.04, 14.44, 4.84}

Variance = $\frac{10.24 + 3.24 + 0.04 + 14.44 + 4.84}{5} = \frac{32.8}{5} = 6.56$

finally take square root to get SD. = $s = \sqrt{6.56} = 2.56$

Sample variance : Square of standard deviation & represents average of the squared diff blw each data point and the mean.

example; variance = $(5)^2$ ⇒ Same data set using calculating sample variance ie. $(2.56)^2 = 6.56$.

Q.9] Explain Quantmod Package.

→ • The Quantmod Package is a popular R Package designed for quantitative financial modelling & trading.

• Quantmod stands for quantitative financial modeling framework has two main fr : download data & charting.

→ install.packages("quantmod")
library(quantmod)

downloading data: use get symbols to get data
get symbols ("AAPL")

- Charting : Quantmod draws Nice charts of following types:
  - line - bar - Candlesticks.

  We can use chartseries() & specify the types directly.

  example:-
  chartseries (AAPL, type = "line", subset = '2007', theme = chartTheme ("white"))

- The R quantmod package make important market based datasets available from several major sources: Google, mySQL, etc...
  Google source (src = "google")
- key feature & functionality of quantmod package.
  - ① Data retrival: facilitates easy retrival of financial data from various various sources, including online DB, APIs & local files.
  - ② Data Manipulating: & ~~transforming data~~.: once imported 'quantmod' offers powerful tool for manipulating & transforming data.
  - ③ Technical Analysis: - Package includes wide ranges of function for conducting technical analysis on financial time series data.
  - ④ Charting & visualization ⑤ Modelling & Strategy development
  - ⑥ integration with other packages ⑦ Educational Resources.

Q.10] Write a short note on sharpe ratio.

→ The sharpe ratio is a widely used measures in finance to evaluate the risks-adjusted return of an investment or a portfolio.
The sharpe ratio helps investors understand the returns they receive for the level of risk they take

Formula for sharpe ratio is: $\frac{E\{R_p\} - \mu_f}{\sigma_p}$,

where R_p → expected return of portfolio

μ_f → risk free rate of return, typically yield on gavernment bonds.

σ_p → standard deviation of portfolio returns volatility or risk

- key points about the sharpe ratio :-

(1) Interpretation : A higher sharpe Ratio indicates a better risk - adjusted return lower sharpe ratio suggest lower return.

(2) Risk - adjustment : By subtracting risk-free rate from portfolio return, the sharpe ratio accounts for the excess return earned above risk free rate

(3) Comparison : Enables investors to compare the risk-adjusted returns of different investment or portfolios.

(4) Utility : Valuable tool for portfolio managers, investors and analyst in accessing efficiency of investment strategies & making informed decisions abat assest allocation and risk management.

(5) Sharpe Ratio provides a concise measure of risk-adjusted return, helping investors quantify the trade-off b/w risk and return and make better-informed investment decisions.

Name :- Sudham Dharmendra Singh

Branch :- CSE (AIML)

Roll no :- AIML57

Subject :- Artificial Intelligence in Finance & Banking System.

Topic :- Assignment No:- (2)

Date of Submission:-

Signature :- Singh

Q.1] Explain K-means Clustering

→ • k-means clustering is a popular unsupervised ML algorithm used for partioning data into clusters.
• In finance & Banking, it can be benefited for customer segmentation, risk analysis, & portfolio optimization.

Algorithm:

k-means iteratively assigns data points to cluster based on their proximity to the cluster centroids.
It aims to minimize the within-cluster sum of squares

Formula:

- Euclidean Distance: $D(x_i, c_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - c_{jk})^2}$

- Cluster Centroid Update: $C_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$

where: n → degree of freedom
Σ → scale matrix

Procedure:

① Initialize centroids randomly.
② Assign each data point to the nearest centroid
③ Update centroids to the mean of the data points assigned to each cluster.
④ Repeat steps ② & ③ until convergence.

Application:

In finance, it can segment customers based on their financial behaviour, group stocks based on similar price movements, or identify patterns in transaction data for fraud detection.

Q.2] Explain Wishart Distribution.

→ The Wishart Distribution is a multiverse vaoiate probability distribution used in statistics to model covariance matrices. In finance & banking, it's often employed in portfolio optimization, risk management, and Bayesian inference.

Defⁿ: The Wishart Distribution characterizes the distribution of the sample covariance matrix of a set of random variables.

Probability Distribution Function [PDF]:-

$$F(x; n, \Sigma) = \frac{|x|^{\frac{n-d-1}{2}} e^{-\frac{1}{2}(tr)(\Sigma^{-1}x)}}{2^{nd/2} (|\Sigma|^{n/2} F_d (n/2)}$$

(or)

$$F(S) = \frac{|x|^{\frac{n-p-1}{2}}}{2^{np/2} \Sigma^{n/2} Fd(n/2)} \cdot \exp\left(\frac{-tr(\Sigma^{-1}x)}{2}\right)$$

where,

$tr(\cdot)$ → trace of matrix	n → degree of freedom
$f_d(\cdot)$ → multivariate gamma function	Σ → scale matrix
x → +ve definete matrix (sample convariance)	

The PDF of the wishart distribution involves the gama function and matrix operation, making it computationaly intensive. It depends on two parameters;

① Degree of freedom (V) & scale } which determines the spread
② Scale matrix (Σ) } and shape of distribution

Application:

It's used in multivarse analysis, such as modelling asset returns in portfolio management or estimating covariance matrices in risk analysis.

Q. 3] Explain Markov Regime Switching Model.

→ • The Markov Regime Switching Model is a time series model that captures changes in the underlying state of system over time.
• In finance, it is used to model regime shifts in asset returns, volatility or economic indicators.

Components include:
① an assumed no. of regimes.
② a dependent variable
③ independent variable
④ Parameters relating dependent variables to independent variable for each regime.
⑤ transition probabilities.
⑥ Statistical influences on model parameters & determined states.

It assumes that the system can be in one of multiple states, and the transition b/w states follows a markov process.

Model consist of main → state dependent parameters &
→ transition probabilities.

Equations:
- state equations: $x_t = \mu_{st} + E_t$ (state-dependent dynamics)
- transition equation: $P_{ij} = P(S_{t+1} = j / S_t = i)$ (transition probabilities)

The model includes state equation to describe evaluation of variables within each state & transition equation to model the switching process.

Application:
It's used to model financial time series with structural breaks, such as bull & bear markets or to capture changes in volatility regimes.

(Q.4) Explain Bayesian Reasoning.

→ Bayesian reasoning is a statistical approach for updating beliefs or making prediction based on prior knowledge and observed data In finance & Banking, it is applied in risk assessment, portfolio management & fraud detection.

It follows Baye's Theorem, which describes how to update the probability of a hypothesis based on new evidences.

Bayes Theorem:-

$$P(H/E) = \frac{P(E/H) \cdot P(H)}{P(E)}$$

where,

- $P(E/H)$ → likelihood of evidence E given hypothesis H.
- $P(H)$ → Prior prob. of hypothesis H.
- $P(H/E)$ → Posterior Prob. of hypothesis H gives evidence E
- $P(E)$ → prob. of evidence E.

Bayesian Reasoning involves specifying prior beliefs, collecting data, updating beliefs using Bayesian theorem, and making decision based on the posterior distribution.

Application:

It's used in credit scoring to update credit risk assessments, in portfolio management to incorporate new info. into asset allocation decisions, and in fraud detection to update fraud probabilities based on transaction data.

Advantages:

It provide a principled framework for decision-making under uncertainty and allows for the incorporation of subjective beliefs and expert knowledge.

Q.5] Explain state machines in market sentiment

→ State Machines in Market Sentiment analysis represent different states & phases of market behaviour based on sentiment indicators. In finance, they are used to model investor psychology, market trends & trading strategies.

A state machine defines a set of states, transition b/w states and action associated with each state.

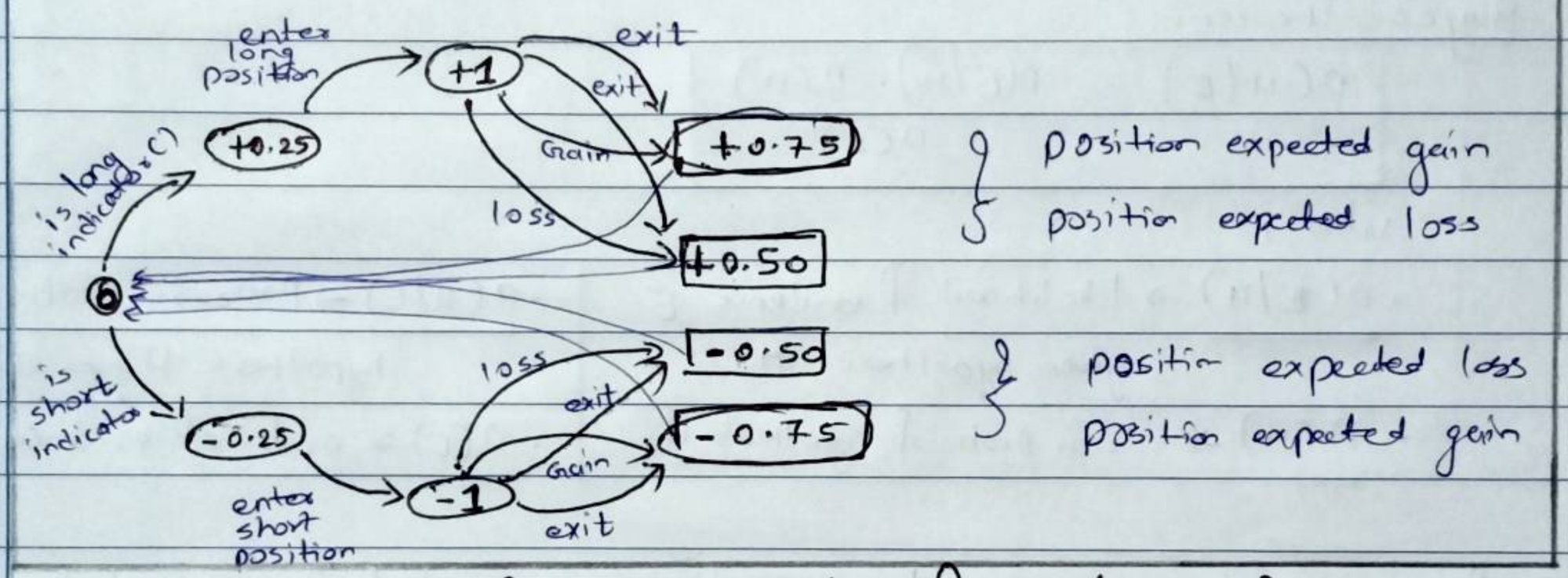


fig. State Machine for long/short Position

- State transition rule:-

$$S_{t+1} = f(S_t, e_t)$$ where; S_t & S_{t+1} are current & next state.
e_t → event triggering state transition
$f(\cdot)$ → transition function.

- States can represent bullish, bearish or neutral market condition based on sentiment indicators like sentiment scores, price movements & trading volumes.
- Transition occur based on predefined rules or condition, such as crossing a threshold or a change in sentiment trend.

Application:- State machine are used in algorithmic trading to trigger buy or sell signals based on market sentiments, in risk management to adjust exposure based on sentiment analysis, and in behavioural finance to study investor sentiment cycles.

Q.6] Explain backtesting an SMA - based strategy.

→ • Backtesting an SMA (simple Moving Average) based strategy involves testing the performance of trading rule based on SMA signals using historical data.

• In Finance, it's used to evaluate the effectiveness of the trend-following strategies.

Explanation:

Strategy - an SMA based strategy generated buy or sell signals based on the crossover of short term & long term moving average.

Backtesting - Backtesting involves applying the strategy to historical data to stimulate trading decision & calculating performance metrices such as returns, sharpe ratio & drawdowns.

Simple Moving Average [SMA]:

$$SMA = \frac{\sum_{i=t}^{t=n} Price_i}{n}$$

{ average closing points (price) of an asset over the last "n" periods ← SMA

Steps: ① Define the short-term & long term windows [SMA Parameters]
② Generate trading signals based on SMA crossovers.
③ Stimulate trading decision [buy, sell or hold] based on signals.
④ Calculate portfolio returns and performance metrics.

Validations:-

Backtesting results should be validated using out-of-samples data & sensitivity analysis to assess robustness.
There is no best or worst simple moving average. you need to backtest to findout what is best time frame for each particular asset.

Q.7] Explain Event based backtesting.

→ Event-based backtesting is a method for evaluating trading strategies based on specific events or triggers in financial markets. It's widely used in algorithmic trading & quantitive finance for strategy development & validation.

Explanation:

Events: It can include price movements, volume, spikes, news announcements or technical indicators crossing predefined thresholds.

Strategy: Event-based strategies generate buy or sell signals based on the occurance of specific events.

Event trigger Rule: $\text{Signal}_t = \begin{cases} 1 & \text{if } E_t \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases}$

Backtesting:

Backtesting involves simulating trading decisions triggered by events using historical data & evaluating the performance of strategy

key-consideration:

- define event types & criteria for triggering signals.
- determine reaction time & trading horizon after an event occurs
- account for transaction costs, slippage, and liquidity constraints in the backtesting process.

Performance metrics:-

Performance metrics such as hitrate, profit factor and average trade duration are used to assess the effectiveness of event-based strategies.

Q.8] Explain the analytical fraud model life cycle.

→ The analytical fraud model life cycle refers to the stage involved in developing, implementing & maintaining a fraud detection model in banking & financial institution.

- Stages:-

① Data collection:- Gather historical data, including fraudulent and non-fraudulent instances.

② Feature Engineering: Extract relevant features from the data, such as transaction amount, frequency, location & behavioral patterns.

③ Model development: build & train ML models, such as logistic regression, random forest (or) neural network using labelled data.

④ Model Evaluation: Assess the performance of the model using evaluation metrics like accuracy, precision, recall & F1 score on a validation dataset.

⑤ Deployment: implement the model into the operational system for real-time fraud detection or batch processing.

⑥ Monitoring: Continuously monitor the models performance retrain it periodically with new data, and update it to accept to evolving fraud patterns.

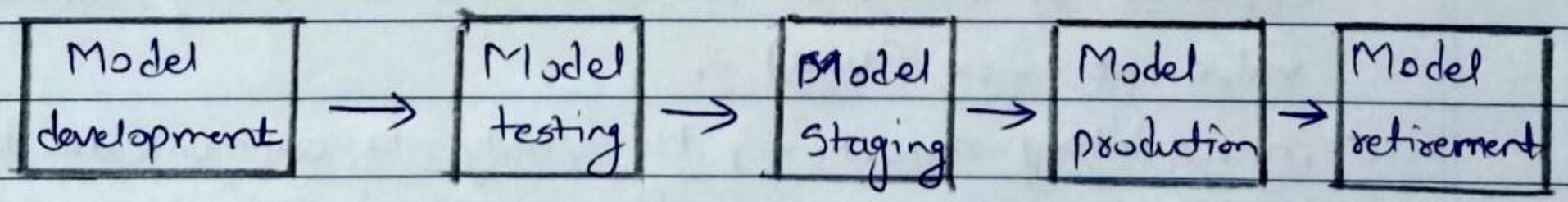


fig. Analytical model life cycle.

Challenges:

Challenges can be in model life cycle include data quality issues, class imbalance, model interpretability & regulatory compliance

Q.9] Explain traffic light indicator approach.

→ The traffic light indicator approach is a risk management technique used to assess and communicate the level of risk associated with traffic activities or events.
In finance, it's applied to in credit risk management assessment portfolio management & regulatory reporting.

Color coding:

The approach categories risk leads using color codes similar to traffic lights: Gain for low risk, yellow for moderate risk and red for high risk.

Criteria:

Risk criteria are pre-defined based on quantitive thresholds, quantitive assessments or a combination of both.

Applications: -

- in credit risk management, it is used to categorize borrowers into risk segments based on credit score, default probabilities and the financial ratios.
- in portfolio management, it's employed to monitor the risk exposure of investment portfolios based on volatility, correlation & a value-at-risk (VaR).
- in regulatory reporting, it's utility to communicate the capital adequacy & solevancy of financial instruction to regulators or stake holders.

• Criteria are pre-defined thresholds or qualitative assessments used to assign risk levels.

Communication: traffic light indicators provide a visual representation of risk levels, facilitating-decision-making & communication among stakeholders

(1).10] Explain backtesting analytical fraud model.

→ Backtesting an analytical fraud model involves assessing its performance using historical data to validates it's effectiveness in detecting fraudulent transaction.

Explanation:-

① Data preparation: Select historical transaction data spanning a select relevant time period, including labelled instances of fraud and non-fraud

② Model application: apply fraud detection model to the historical data to generate prediction or scores indicating the likelihood of fraud for each transaction.

③ Performance evaluation: compare the model's prediction against the actual bias labels to calculate performance metrics such as accuracy, precision, recall & F-1 score.

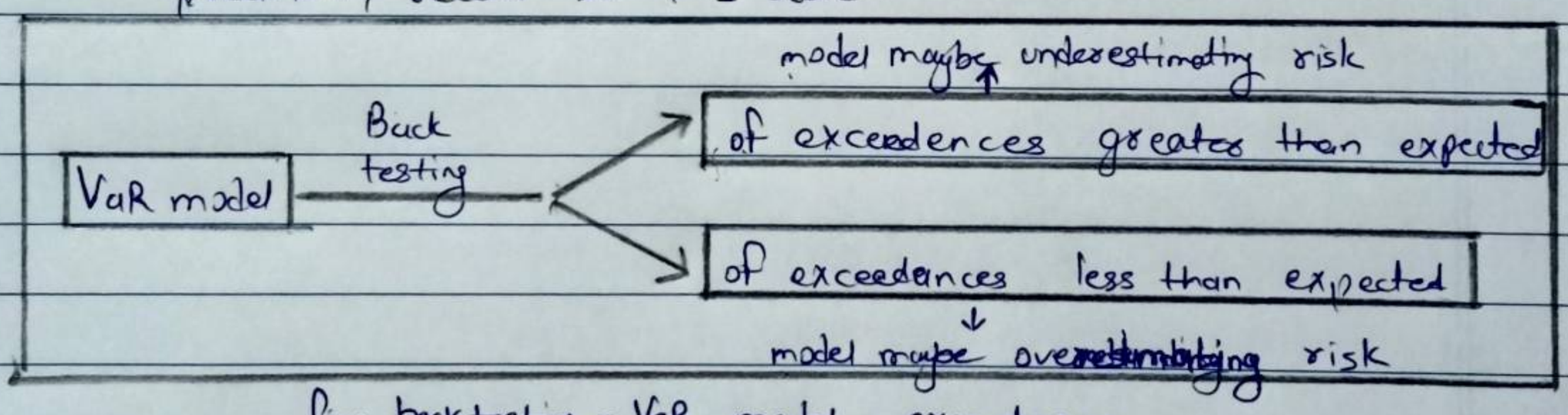


fig. backtesting - VaR - model - exceedances,

④ Validation:- Validate the model's performance using out-of-samples data coo) cross-validation techniques to ensure robustness & generalizatn

⑤ Analysis:- Analyzes the result, to identify areas of improvement, assess the impact of FP & FN & refine model's parameters or features.

⑥ Documentation:- Document the backtesting process, including data sources, model specification, performance metrics & validation results, to comply with regulatory requirements & internal policies.