



**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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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_expl.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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
SUBJECT	AI FOR FINANCE & BANKING APPLICATION LAB
COURSE CODE	CSDOL8011
PRACTICAL NO.	
DOP	
DOS	



Code:

```
[3] 0s 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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
SUBJECT	AI FOR FINANCE & BANKING APPLICATION LAB
COURSE CODE	CSDOL8011
PRACTICAL NO.	
DOP	
DOS	



Output:

```
[✓ 2s] [1] import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt
      from sklearn.cluster import KMeans
      import warnings
      warnings.filterwarnings('ignore')

[✓ 0s] [8] df = pd.read_csv(r"mall_customers.csv")

[✓ 0s] [9] df.head()

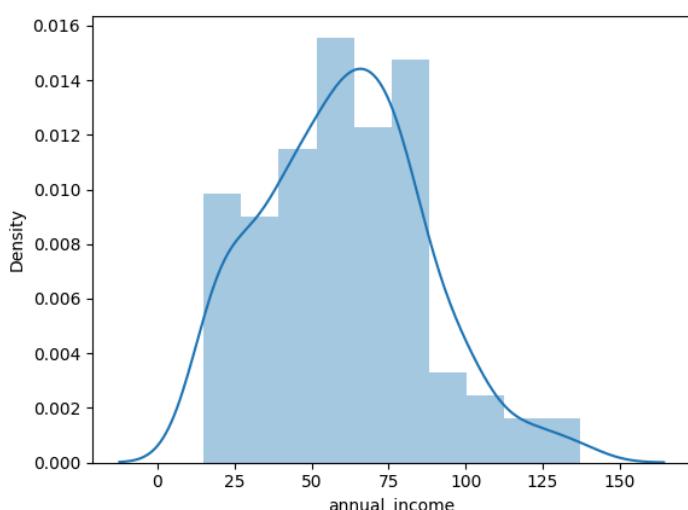
  customer_id  gender  age  annual_income  spending_score
0            1    Male   19             15              39
1            2    Male   21             15              81
2            3  Female   20             16               6
3            4  Female   23             16              77
4       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

Next steps: View recommended plots

[✓ 0s] [10] df.describe()

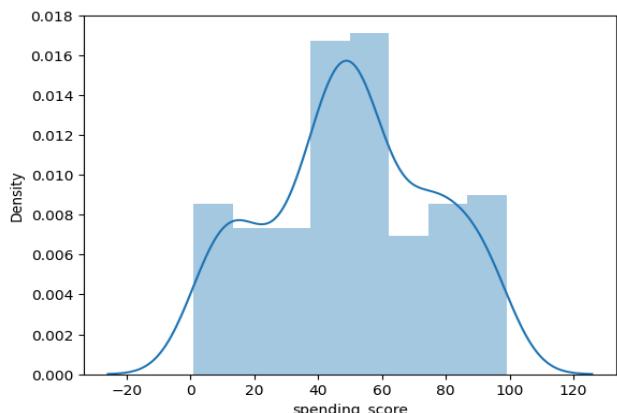
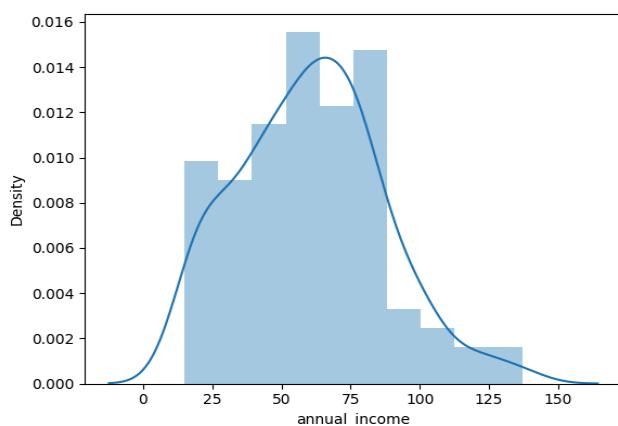
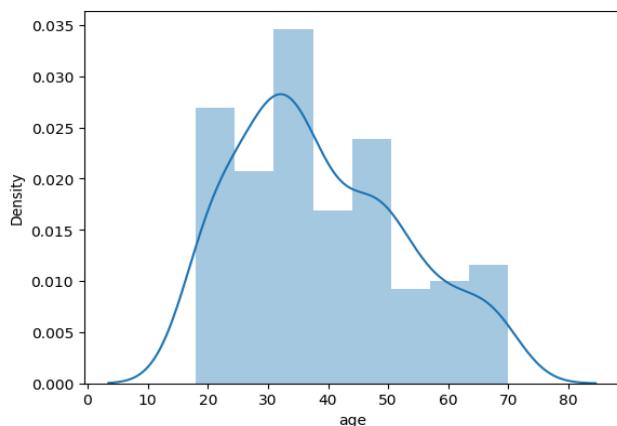
  customer_id  age  annual_income  spending_score
count    200.000000  200.000000  200.000000
mean     100.500000  38.850000  60.560000
std      57.879185  13.969007  26.264721
min      1.000000  18.000000  15.000000
25%     50.750000  28.750000  41.500000
50%     100.500000  36.000000  61.500000
75% 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

```
[✓ 1s] [12] sns.distplot(df['annual_income']);
```

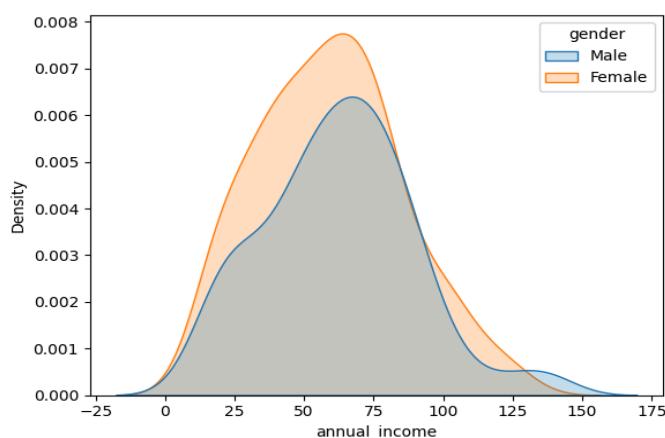


```
0s [13] df.columns
Index(['customer_id', 'gender', 'age', 'annual_income', 'spending_score'], dtype='object')
```

```
2s [15] columns = ['age', 'annual_income', 'spending_score']
for i in columns:
    plt.figure()
    sns.distplot(df[i])
```



```
0s [20] sns.kdeplot(data=df, x='annual_income', shade=True, hue='gender')
```



```
0s [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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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()
```



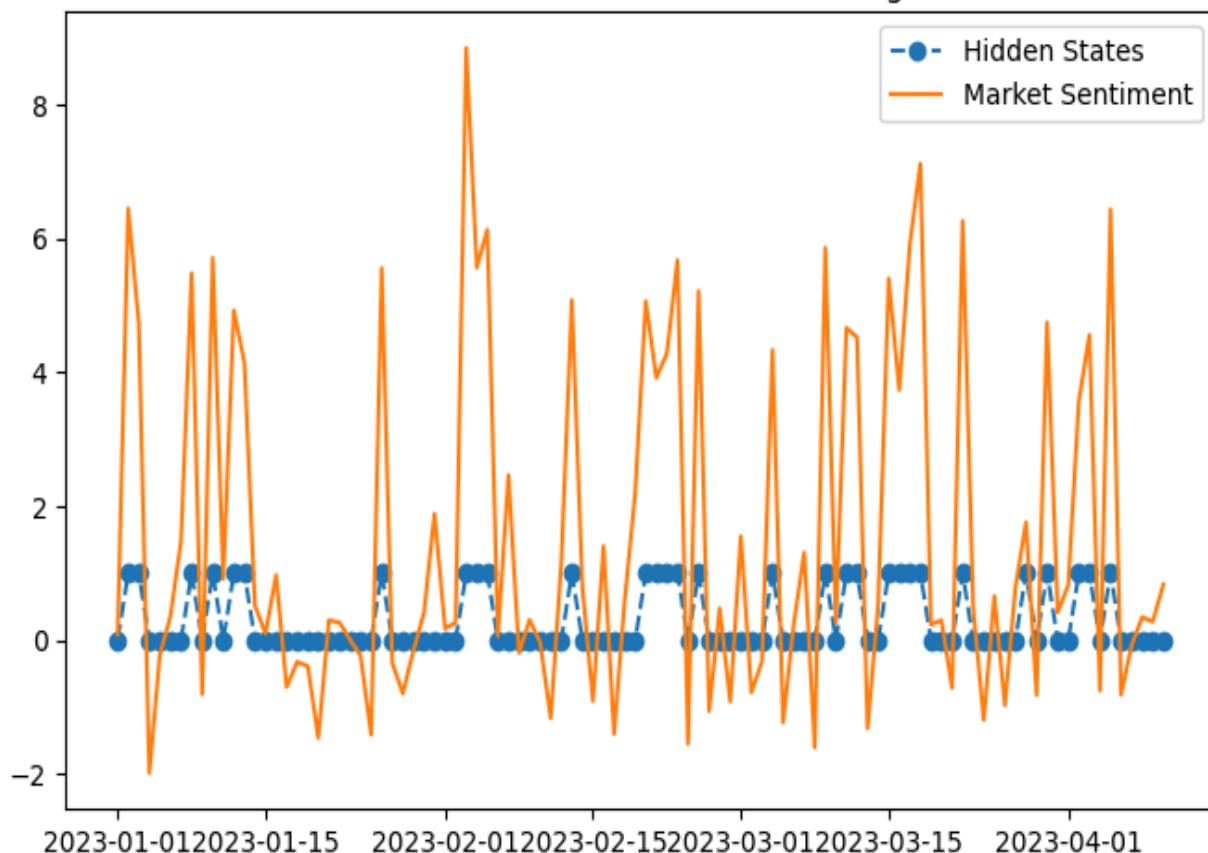
Output:

```
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 - 04-10-2023    HQIC: 514.549
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.
```

Market Sentiment with Markov Switching Model





**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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
SUBJECT	AI FOR FINANCE & BANKING APPLICATION LAB
COURSE CODE	CSDOL8011
PRACTICAL NO.	
DOP	
DOS	



Output:

```
✓ 0s [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')

✓ 0s [2] url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'

✓ 0s [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
```

```
✓ 1s [5] In [3]: data['SMA1'] = data[symbol].rolling(42).mean()
     data['SMA2'] = data[symbol].rolling(258).mean()
     data.plot(figsize=(10, 6));
```



```
✓ 2s [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');
```



```

✓ 0s [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

✓ 0s [8] data[['r', 's']].sum().apply(np.exp) - 1

      r    -0.1466
      s    0.4075
      dtype: float64

✓ 1s [9] data[['r', 's']].cumsum().apply(np.exp).plot(figsize=(10, 6));

```



```

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

      10

✓ 0s [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))
```



```
[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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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-fraud)
# 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_size=0.2, random_state=42)

# Create a Random Forest Classifier model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# 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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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()

```

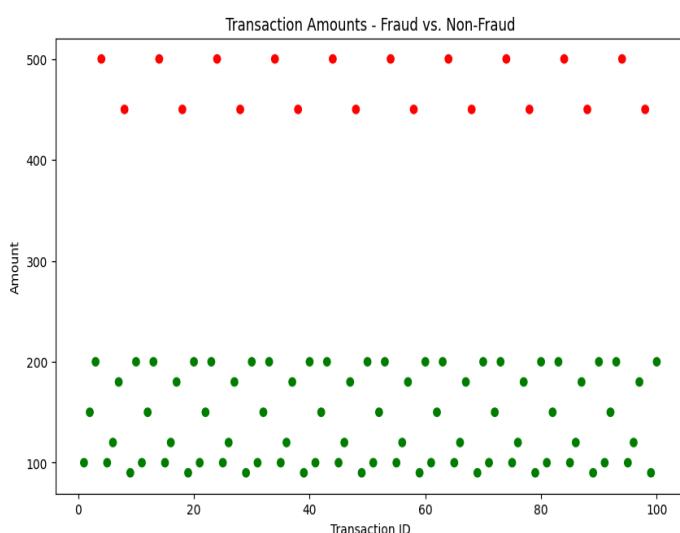


Fig. 1



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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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'],
        'Returns': returns
    })

    return portfolio
```



```
'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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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)

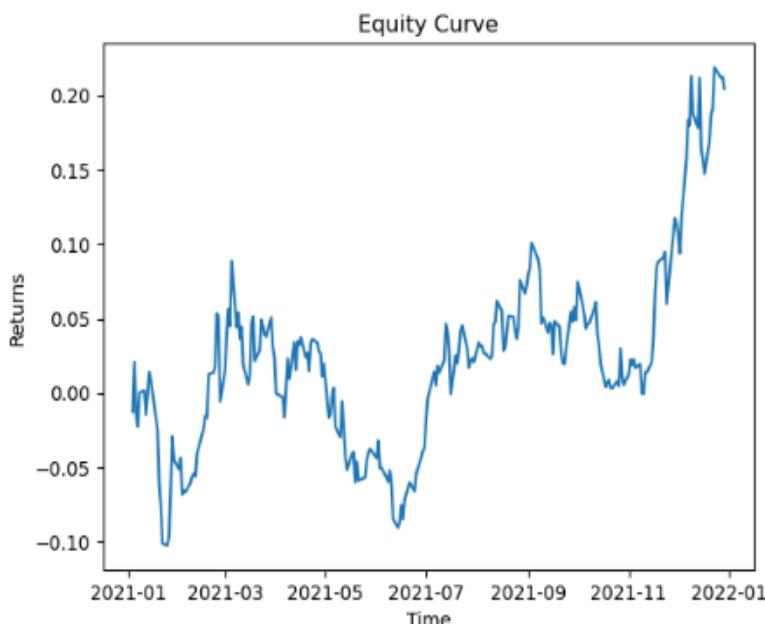
print("Sharpe Ratio:", sharpe_ratio)
print("Max Drawdown:", max_drawdown)

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	PRATHAMESH S. CHIKANKAR
BRANCH	CSE-(AI&ML)
ROLL NO.	AIML11
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

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

[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)

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

RandomForestClassifier

RandomForestClassifier(random_state=42)

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

	actual	prediction	grid icon
6252	0	0	grid icon
4684	0	0	grid icon
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:- Prathamesh S. Chikankar

Roll No:- Aim11 Year:- BE Sem VIII

Branch:- (SE-CIVIL) Subject:- AI for finance and
Banking application
(AI FB)

Topic:- Assignment No. 02

Date:- Feb'24

Sign:- Prathamesh



1. Explain benefits of BCt in the Banking sector.

- 1) costs reduced: one of the benefits of Blockchain for Banks is reduced costs. Banks have recently learned that blockchain can allow them to reduce infrastructure costs upto \$20 billion by year 2022. Banks can reduce cost b/w bank to Bank transaction. (smart contract)
- 2) faster transactions: any transaction can be done within a matter of seconds and slightly faster than other traditional methods.
- 3) improved security: shared ledgers can help banks better secure transaction information. Two security keys exist for each transaction. Data is unchangeable after verified.
- 4) improved data quality: Blockchain technology can store any type of data and allows it to be accessed following predefined rules and regulations. Smart contracts automatically verifies & approves contracts.
- 5) Digital currencies: Banks can benefit from blockchain with 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 assets.
- 7) Reduced Transactional Time: Allowing more easily reconcile transaction. Trace transaction more quickly & find error in a timely manner.

2. Write a short Note on GIFT City.

GIFT city, which stands for Gujarat International Finance Tec-City is a planned business district in the Indian state of Gujarat. The project is intended to promote international financial services and create a business-friendly environment. It is designed to be a global financial and IT services hub and is located b/w Ahmedabad and 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) FinTech Hub: providing a platform for various activity.
- 3) Infrastructure and connectivity: designed with modern infra., including high-quality offices space & connectivity through road.
- 4) International Exchange: Hence, the IFSC, includes international exchange for trading various financial instruments & currencies.
- 5) Regulatory framework: within Indian legal system (framework).
- 6) Smart city features: enhance efficiency & sustainability.
- 7) Global companies & institutions 8) Government support.

3)

Explain digital money transfer mechanisms.

Involves electronic movement of funds from one party to another.

- 1) Online Banking: Bank transfer - within same bank (money transfer) & inter-transfer - Electronic funds transfer (Bank to Bank).
- 2) Mobile Banking Apps: peer-to-peer (P2P) Payments - user can transfer money directly to friends @ family through mobile apps.
mobile wallets - example. Google pay, Apple pay.
- 3) Digital wallets: e-wallets - platforms like paytm, vimo.
Cryptocurrency wallets: Cryptocurrency like Bitcoin → digital wallets.
- 4) peer-to-peer (P2P) platforms: P2P lending & platform-sure platform enable users to lend and borrow money directly from one another.
- 5) Cryptocurrencies transfer: BC7 - ensuring secure & transparent transaction.
- 6) Cross-Border payment services: services like western union.
- 7) Digital Banks: Neobanks - being digital banks.
- 8) Payment Gateways: Online merchant-platform like stripe, square.
- 9) QR code payments: Scanning QR codes - Google pay.
- 10) Central Bank Digital Currency: (CBDCs): Some country are employing & implementing their own currency issued by Central Bank.



4. Explain application of AI in Finance & Banking Sector.
- 1) Cybersecurity and Fraud detection: AI-powered cyber-security systems analyze network traffic and user behavior to detect and prevent cyber threats & identify fraudulent transactions.
 - 2) Customer experience and Chatbots: AI-driven chatbots provide personalized assistance, answering customer queries.
 - 3) Risk management: assesses various risk factors, including market volatility and creditworthiness & minimize losses.
 - 4) Loan & credit decision process: Analyze diverse data sources to evaluate loan applications, accurately assessing borrowers' creditworthiness & streamlining loan approval process.
 - 5) Trade & market trends: analyze large volume of market data in real-time to identify trends, patterns, and opportunities.
 - 6) Data collection and analysis: AI systems collect and analyze vast amount of data structured & unstructured from diverse sources, providing insights into consumer behaviour.
 - 7) Predictive analysis in regulatory compliance: industry leverages AI and ML to anticipate potential compliance risks, identify process patterns of non-compliance & predict regulatory changes.

5. Explain digitization in the financial & Banking sector.
- Reshaping traditional processes, enhancing customer experience and improving overall efficiency.
- 1) Digital Banking: online Banking - customer can access their accounts, transfer funds & transact through secure online portals, mobile Banking - accessible via mobile devices including smartphones.
 - 2) Cloud-Based core Banking: core processing systems. provide centralized system / platform for managing essential functions of Banking such as deposits, loans, and insurance.



- 3) Customer relationship management (CRM): personalized service - CRM system help in understanding customer preference, enabling banks to offer personalized services & tailors marketing.
- 4) Data Analytics and Business intelligence: customer insights - utilizing analytics to gain valuable insights into customer behavior & risk management - predictive analysis helps banks.
- 5) Cybersecurity solutions: cloud-based security - utilizing cloud-based security services to protect sensitive financial data from cyber threats.
- 6) APIs (application programming interface): open banking - APIs facilitate secure data sharing between financial institutions, enabling the development of innovative financial products & services.
- 7) Digital wallets & payments
- 8) Blockchain & ledger
- 9) AI-powered chatbots
- 10) Digital lending platforms.

6. Explain the skewness and kurtosis with formulas?

Skewness: Skewness measures the asymmetry of the distribution of data around its mean. Skewness of zero indicates a symmetric distribution, while the +ve skewness indicates a skewed to right (left respectively).

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

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

In R, the skewness (sk) is found moments library with moment method above to compute the sample skewness, where n represents no. of observations in sample.

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

s → sample standard deviation.

Kurtosis: measuring the degree of peakedness (flatness) of a distribution.



(compared to Normal distribution). positive kurtosis indicates a sharper peak (leptokurtic), while negative kurtosis indicate flatter peak (platykurtic) compared to normal distribution.

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

$$\text{Kurt}(R) = \frac{1}{N} \sum_{i=1}^N \left\{ \frac{(R_i - \bar{R})}{S} \right\}^4$$

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

7. Explain sample covariance and correlate with formulas?

Sample Covariance: measures the degree to which two variables change together. A positive covariance indicates that the variable tend to increase \Rightarrow decreases together, while negative covariance indicates that one variable tends to increase as other decreases.

defined as, $\Sigma = \text{Cov}(R_j, R_k) = E \{ (R_j - \bar{R}_j)(R_k - \bar{R}_k) \}$

where, "E" → Expected value due to random variable

also, $\bar{R}_j = \frac{1}{N} \sum_{i=1}^N R_{ij}$ and $\bar{R}_k = \frac{1}{N} \sum_{i=1}^N R_{ik}$.

Correlation: Sample correlation is a standardized measure of strength and direction of linear relationship between two variables. It is obtained by dividing sample covariance by the product of the sample standard deviation of the two variables.

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

correlation coefficient ranges b/w -1 and 1.

8. Explain sample mean, standard deviation, and 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 & turn it into no. of observations.

Example: Consider dataset for exam score: {85, 90, 88, 92, 86}.

the sample mean \bar{R} can be calculated by: $\bar{R} = \frac{1}{N} \sum_{i=1}^N R_i$

$$\bar{R} = \frac{85 + 90 + 88 + 92 + 86}{5} = \frac{441}{5} = 88.2 \quad \leftarrow \text{sample mean of dataset.}$$

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

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (R_i - \bar{R})^2} \quad \left\{ \begin{array}{l} \text{using some dataset/queue set} \\ \text{example, calculate std. b/w each score.} \end{array} \right.$$

and the mean is $\bar{R} = 3.2, 1.8, 0.2, 3.8, -2.2$ } square of each diff

$$\text{standard deviation} = \sqrt{10.24 + 3.24 + 0.04 + 14.44 + 4.84} = \frac{32.8}{5} = 6.56$$

binary table square root to get $s.d.: s = \sqrt{6.56} \approx 2.56$

Sample variance: square of standard deviation and represents average of the squared diff b/w each data point and the mean.

$$\text{example, variance} = (s)^2 \quad \left\{ \begin{array}{l} \text{same dataset using calculate} \\ \text{sample variance formula} (2.56)^2 \Rightarrow 6.56 \end{array} \right.$$

Explains about packages used for analysis:

The "quantmod" package is a popular R package designed for quantitative financial modeling and trading.

Quantmod stands for quantitative financial modeling framework has two main fn: download data and charting.

→ install.packages ("quantmod")

library (quantmod)

downloading (data): use "getSymbols" to get data.



`getSymbols("AAPL")`

> chartSesg: Quarture chart shows rule charts of following types:

- candlestick - bar - line

We can use `chartSesg()` & specify the types directly.

Example: `chartSesg("AAPL", type = "bar", subscr = '2007', theme = chartTheme`

`white")`

> The R quantmod package makes important market-based

datasets available from several major sources: Google, MySQL, etc.
(Google Source (src = "google"))

> Key features & functionality of 'quantmod' package:

1) Data retrieval: facilitate easy retrieval of financial data from various sources, including online DB, APIs & local files.

2) Data manipulation: Once imported, 'quantmod' offers powerful tool for manipulating and transforming data.

3) Technical analysis: package includes wide range of tools for conducting technical analysis on financial time series data.

4) charting and visualization 5) integration with other packages

6) modeling & strategy development 7) Educational resources.

10. Write a short note on Sharpe ratio.

The Sharpe ratio is a widely used measure in finance to evaluate the risk-adjusted return of an investment in a portfolio.

The Sharpe ratio helps investors understand the return they receive for the level of risk they take.

Formula for Sharpe ratio is: $\frac{E[R_p] - R_f}{\sigma_p}$, where,

$R_p \rightarrow$ expected return of portfolio

$R_f \rightarrow$ risk-free rate of return, typically yield on government bonds.

$\sigma_p \rightarrow$ standard deviation of portfolios return, volatility. \Rightarrow risk

\triangleright 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 the risk-free rate.

3) Comparison: Enables investors to compare their risk-adjusted returns of different investments \Rightarrow portfolios.

4) Utility: valuable tool for portfolio managers, investors and analysts in assessing the efficiency of investment strategies & making informed decisions about asset allocation and risk management.

\triangleright Sharpe Ratio provides a concise measure of risk-adjusted return, helping investors quantify the trade-off between risk and return and make better-informed investing decisions.

of the superior return (higher Sharpe Ratio) is given by the formula:

$$\text{Sharpe Ratio} = \frac{\text{Expected Return} - \text{Risk-Free Rate}}{\text{Standard Deviation}}$$

where Expected Return = $E(R_p)$ and Standard Deviation = σ_p .

After square root of Sharpe Ratio is taken

function of standard deviation is minimized if Sharpe ratio is 0.09 or 0.10. This function is of minimum deviation - the best portfolio and minimum standard deviation of a given Sharpe ratio.

Best point state for which we get minimum standard deviation, $\sigma_p = 5.94\%$

square of Sharpe ratio of maximum

90

square of Sharpe ratio = 0.98

and maximum Sharpe ratio, which is our objective - i.e.



Name: Prathamesh B. Chikankar

ROLL NO.: A3ML11 YEAR: BE

Branch: CSE-[A18M9L] SERG: VIII

Subject: Artificial Intelligence for Finance and
Banking applications (AI(FBA))

Topic: ASSIGNMENT NO. 02

Sign: Prathy

Date: April '24



1.

Explain K-means clustering.

K-means clustering is a popular unsupervised ML algorithm used for partitioning data into clusters.

In finance & banking, it can be used 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: $n \rightarrow$ degree of freedom

$\Sigma \rightarrow$ scale matrix

Procedure:

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

Applications:

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

2. Explain Wishart Distribution:

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

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

Probability distribution function (PDF):

$$f(x; n, \Sigma) = \frac{1}{2^{\frac{n(p-1)}{2}} \Gamma_p(\frac{1}{2})} e^{-\frac{1}{2} \text{tr}(\Sigma^{-1} x)}$$

where;

$x \rightarrow$ true definite matrix (sample covariance matrix)
 $n \rightarrow$ degree of freedom
 $\Sigma \rightarrow$ scale matrix

$$\textcircled{(2)} \quad f(x) = \frac{1}{2^{\frac{n(p-1)}{2}} \Gamma_p(\frac{1}{2})} \exp \left(-\frac{\text{tr}(\Sigma^{-1} x)}{2} \right).$$

$\rightarrow \text{tr}(\cdot) \rightarrow$ trace of matrix

$\Gamma_p(\cdot) \rightarrow$ multivariate gamma function.

The PDF of the Wishart distribution involves the gamma function and matrix operation, making it computationally intensive.

It depends on two parameters:

- | | |
|--|--|
| ① degree of freedom (n) & scale
② scale matrix (Σ) | which determines the spread and shape of distribution. |
|--|--|

Applications:

It is used in multivariate analysis, such as modelling asset returns in portfolio management @ estimating covariance matrices in risk analysis.



3.

Hidden Markov Regime Switching model.

The markov Regime switching model is a time series model that captures change in the underlying state of system over time. In finance, it's used to model regime shift in asset returns, volatility & economic indicators.

Components include:

- ① an assumed no. of regimes
- ② a dependent variable
- ③ independent variable
- ④ parameters relating dependent variables to independent variables for each regime.
- ⑤ transition probability
- ⑥ statistical inferences on model parameters & determined state

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

Model consists of main \rightarrow state-dependent parameters and \rightarrow transition probabilities.

Equations:

1) State equations: $x_t = u_s + \epsilon_t$ (state-dependent dynamics)

2) Transition eqn: $P_{ij} = P(S_{t+1} = j | S_t = i)$ (transition probability)

The model includes state eqn to describe evolution of variables within each state & transition eqn to models the switching process.

Applicability:

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



4.

Explain Bayesian Reasoning

Bayesian reasoning is a statistical approach for updating beliefs @ making prediction based on prior knowledge and observed data.

In finance and banking, it's applied in risk assessment, portfolio management, and fraud detection.

It follows Bayes theorem, which describes how to update the probability of a hypothesis based on new evidence.

Bayes theorem:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} \quad \text{where;}$$

$\cdot P(H|E)$ → posterior prob. of hypothesis H giving evidence E

- $P(E|H)$ → likelihood of evidence E given hypothesis H
- $P(H)$ → prior prob. of hypothesis H
- $P(E)$ → 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.

Applictn: It's used in credit scoring to update credit risk assessment, in portfolio management to incorporate new info. into asset allocation decisions, and in fraud detection to update fraud probability 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.



5.

Explain state machine in market sentiment.

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

A state machine defines a set of states, transition rules between states and actions associated with each state.

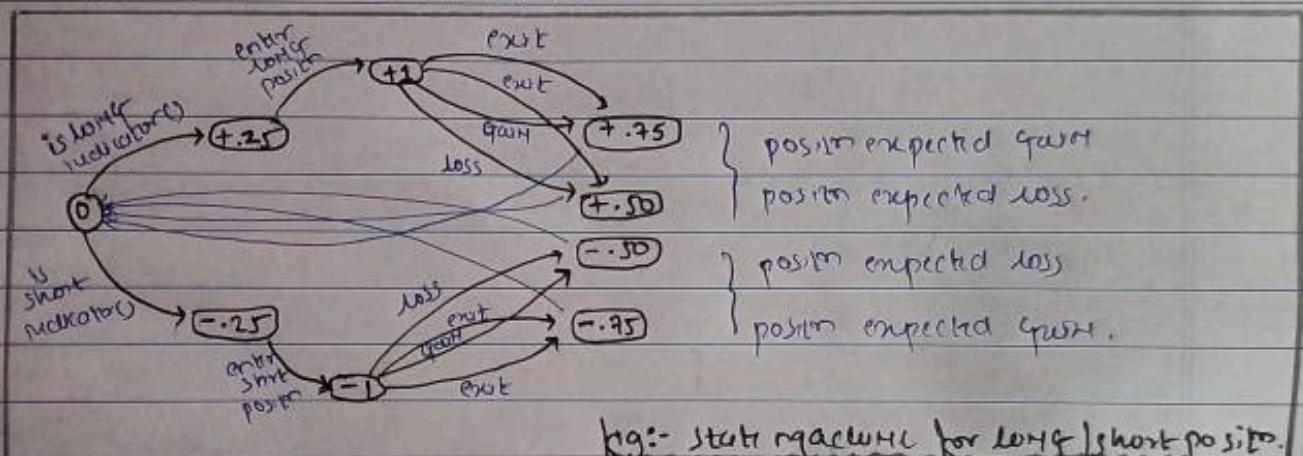


fig:- State machine for long/short position.

> state transition rule:

$$S_{t+1} = f(S_t, e_t)$$

where;

S_t & S_{t+1} are current & next state.

$e_t \rightarrow$ event triggering state transition

$f(\cdot) \rightarrow$ transition fn.

> states can represent bullish, bearish or neutral market condition based on sentiment indicators like Sentiment Scores, power requirements or trading volumes.

> transitions occur based on predefined rules or rules, such as crossing a threshold or a change in sentiment trend.

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

6.

Explain backtesting on 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 by \oplus sell signals based on the crossover of short-term and long-term moving average.

Backtesting: Backtesting involves applying the strategy to historical data to simulate trading decisions & calculate performance metrics like return, sharpe ratio and drawdowns.

Simple moving average (SMA):

$$\text{SMA} = \frac{\sum_{t=1}^{t=n} \text{price}}{n}$$

} average closing price of an asset over the last "n" periods. \leftarrow SMA

Steps:

- ① determine short-term & long-term windows (SMA parameters)
- ② generate trading signals based on SMA crossovers.
- ③ simulate trading decisions (buy, sell \oplus hold) based on signals
- ④ calculate portfolio returns and performance metrics.

Validation:

Backtesting results should be validated using out-of-sample data and sensitivity analysis to assess robustness.

There is no best \oplus worst simple moving average. You need to backtest to find out what is best range for each particular asset.



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 is widely used in algorithmic trading and quantitative finance for strategy development and validation.

Explanation:

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

Strategy: Event-based strategies generate buy or sell signals based on the occurrence 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 and evaluating the performance of the strategy.

Key Considerations:

- > Define event types & conditions for triggering signals.
- > Determine entry logic & trading position sizing on event occurs.
- > Account for transaction costs, slippage, and liquidity constraints in the backtesting process.

Performance Metrics:

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

8.

Explain the analytical fraud model life cycle.

The analytical fraud model life cycle refers to the stages involved in developing, implementing, and maintaining a fraud detection model in banking & financial institutions.

• Stages:

- (1) data collection: Gather historical transaction data, including fraudulent and non-fraudulent instances.
- (2) feature engineering: Extract relevant features from the data, such as transaction amount, frequency, location & behavioural patterns.
- (3) model development: build & train ML models, such as logistic regression, random forest, etc. initially using labelled data.
- (4) model evaluation: Assess the performance of the model using evaluation metrics like accuracy, precision, recall and F1 score on a validation dataset.
- (5) deployment: Implement the model in to the operational system for real-time fraud detection (batch processing).
- (6) monitoring: continuously monitor the models performance, retrain it periodically with new data, and update it to adapt to evolving fraud patterns.

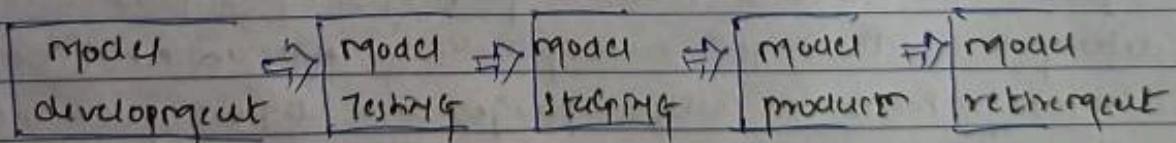


fig:- Analytical model life cycle

challenges: challenges can be in model life cycle include data quality issues, class imbalance, model interpretability and regulatory compliance.



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 specific activities @ events.

In finance, it is applied in credit risk management assessment, portfolio management and regulatory reporting.

Color coding:

The approach categorizes risk levels using color codes similar to traffic lights: green for low risk, yellow for moderate risk and red for high risk.

Criteria:

Risk criteria are pre-defined based on quantitative thresholds, qualitative assessment, @ a combination of both.

Applications:

- > In credit risk management, it is used to categorize borrowers into risk segments based on credit scores, default probability and the financial ratios.
- > In portfolio management, it is employed to monitor the risk exposure of investment portfolios based on volatility, correlation and a value-at-risk (VaR).
- > In regulatory reporting, its ability to communicate the capital adequacy & solvency of financial institutions to regulators @ stakeholders.
- Criteria are pre-defined thresholds @ qualitative assessment used to assign risk levels.

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



10.

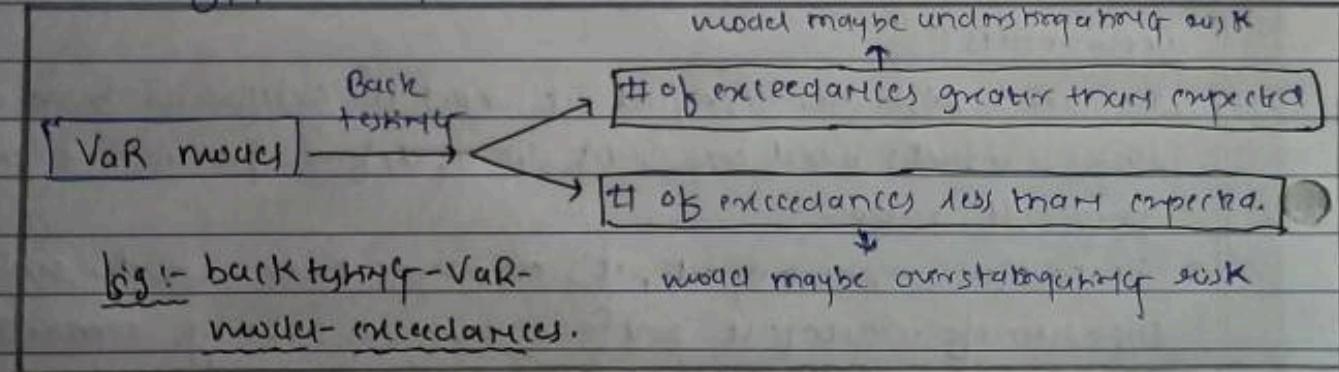
Explain back testing for analytical fraud model.

Back testing an analytical fraud model involves assessing its performance using historical data to validate its effectiveness in detecting fraudulent transactions.

Explanation:

- (1) data preparation: select historical transaction data spanning a selected relevant time period, including labelled instances of fraud and non-fraud.
- (2) model application: apply fraud detection model to the historical data to generate predicted scores indicating the likelihood of fraud for each transaction.
- (3) performance evaluation: compare the model's predictions against the actual bias labels to calculate performance metrics such as accuracy, precision, recall and F-1 score.

model may be underfitting work



- (4) validation: validate the model's performance using out-of-sample data @ cross-validate techniques to ensure robustness & generalization.
- (5) Analysis: analyze the results to identify areas of improvement, assess the impact of FP & FN, refine model's parameters @ features.
- (6) Documentation: document the backtesting process, including data sources, model specification, performance metrics, and validation results, to comply with regulatory requirements and internal policies.