An Empirical Study on the Correlation Between 24K Gold and the Nifty 50 Index

Submitted in partial fulfillment of the requirement for the award of Degree of Masters in Business Administration in Technology Management Discipline

Submitted To

SVKM's NMIMS,
Mukesh Patel School of Technology Management & Engineering,
Mumbai Campus

Submitted by:

Nishant Parwani

Under The Supervision of:

Prof. Radhika Patil
Department Of Computer Engineering

DEPARTMENT OF COMPUTER SCIENCE
Mukesh Patel School of Technology Management & Engineering
ACADEMIC SESSION: 2025-26

CERTIFICATE

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1
Introduction

1.1 Background of the Project Topic

It is common in about all Indian household that gold is financial security. It is usual instinct to save gold as savings for future. While recent upsurge in equity trade doesn't stand unknown. Gold and equities have long occupied a dynamic tug-of-war in financial markets. 24-K gold, in particular, has served as a "safe-haven" whenever economic uncertainty, geopolitical flare-ups, or market turbulence arise. In contrast, the Nifty 50—India's flagship equity index—tracks the performance of the nation's leading companies and is widely regarded as a barometer of economic health. Gold's security can easily be witnessed by reviewing trends in monetary stability of country in geopolitics over the years.

For investors operating in India, this duality takes on added importance. Gold is not only a financial asset or a commodity; it is also woven into cultural traditions, which amplifies its demand during times of stress. To bring figures into perspective this years Dhanteras- A Hindu sub festival saw a massive sale of gold ornaments even after intense escalation of prices due to geopolitical tensions/ Traditional wisdom holds that gold and equities should move inversely, offering natural hedging. Yet the reality is far more diversed—correlations ebb and flow, driven by macro-economic shocks, policy shifts, and global market integration.

1.2 Motivation and Scope of the Report

The genesis for this study is the need for empirically grounded investment strategies tailored to the Indian market. Our key objectives are:

1. Quantify the correlation structure between 24-K gold and the Nifty 50.
2. Answer questions like:
    - *Can movements in the Nifty 50 forecast subsequent gold price changes?*
    - *Translate findings into practice: How can these insights inform portfolio construction and risk management?*

By addressing these questions, we aim to bridge the gap between theory and practice for Indian investors.

1.3 Problem Statement

To what extent can daily movements in the Nifty 50 index predict changes in 24-K gold prices, and what are the implications for investment strategy formulation in India?

1.4 Salient contribution

- Temporal Span: Daily price data from January 2015 to January 2025.
- Methodological Breadth: Combination of linear and non-linear (machine-learning) models to capture both static and evolving relationships.
- Practical Focus: Findings are framed to aid real-world portfolio decisions, rather than purely academic exploration.

Limitations:

- Exclusion of macro-economic variables (e.g., inflation, interest rates).
- Outliers in price of gold or nifty due to geopolitical tensions like war.
- Analysis limited to daily frequency, potentially overlooking intraday dynamics.
- Focus remains on the correlation—other structural factors are not examined.

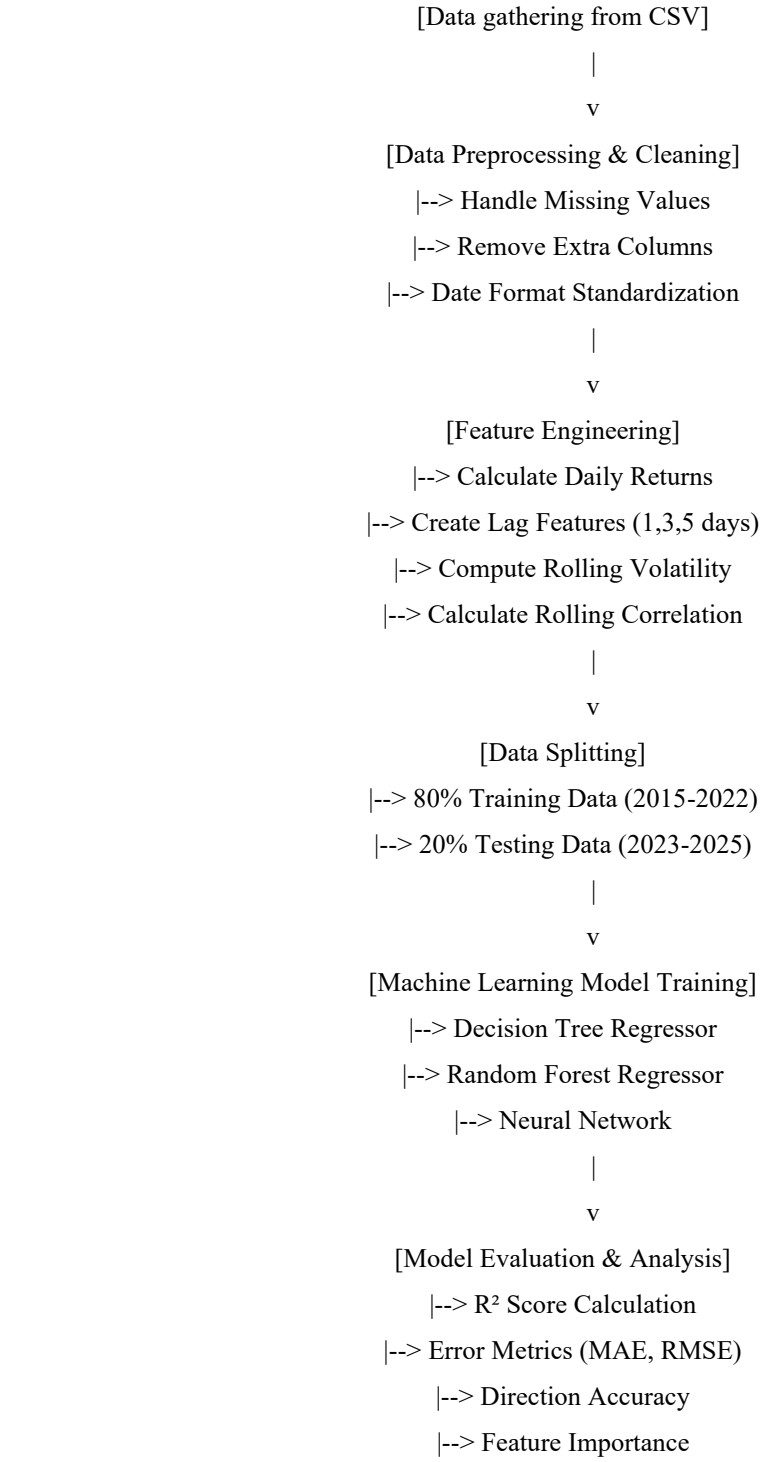
1.5 Organization of report

1. Literature Review: Foundations and previous findings on gold-equity dynamics.
2. Data & Methodology: Description of data sources, preprocessing steps, and analytical tools.
3. Empirical Results: Statistical correlation analysis, predictive modeling outcomes.
4. Discussion & Practical Implications: Interpreting results for portfolio construction and risk management.
5. Conclusion & Future Work – Summarizing insights and proposing avenues for further research.

Chapter 2

Methodology and Implementation

2.1 Block diagram



2.2 Hardware description

The analysis was performed on a standard computing system with the following specifications (for reference only):

Processor: Intel Core i7 CPU

Memory: 16GB RAM

Storage: 1TB SSD

Operating System: Windows 11

Additional Requirements: Internet connectivity for library installation

2.3 Software description, flowchart / algorithm

Programming Language: Python 3.X.X

Key Libraries:

- pandas
- numpy
- scikit-learn
- tensorflow
- matplotlib & seaborn
- CSV
- Regression/Correlation

Development Environment: Visual Studio Code

START

|

├─ Load Dataset "Final DEP Project.csv"

| ├─ Columns: date, open_x, high_x, low_x, close, price, open_y, high_y, low_y, volume, chg%

| └─ Remove Unnamed columns

|

├─ Data Preprocessing

| ├─ Convert date to datetime format

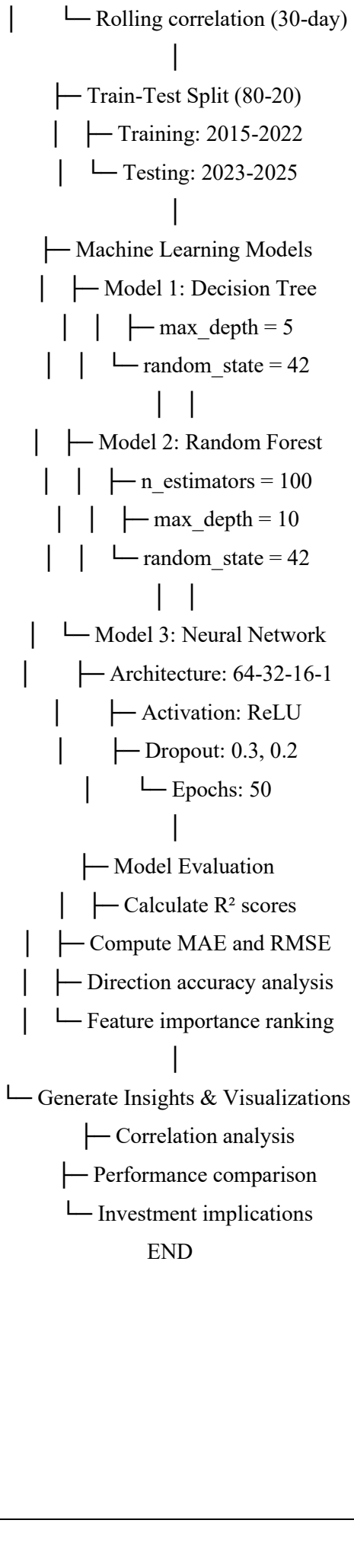
| ├─ Sort chronologically

| ├─ Calculate daily returns for Gold and Nifty

| └─ Create feature set:

| ├─ Lagged returns (1,3,5 days)

| ├─ Rolling volatility (5-day)


└─ Rolling correlation (30-day)
├─ Train-Test Split (80-20)
├─ Training: 2015-2022
└─ Testing: 2023-2025
├─ Machine Learning Models
├─ Model 1: Decision Tree
├─ max_depth = 5
└─ random_state = 42
├─ Model 2: Random Forest
├─ n_estimators = 100
├─ max_depth = 10
└─ random_state = 42
└─ Model 3: Neural Network
├─ Architecture: 64-32-16-1
├─ Activation: ReLU
├─ Dropout: 0.3, 0.2
└─ Epochs: 50
├─ Model Evaluation
├─ Calculate R² scores
├─ Compute MAE and RMSE
├─ Direction accuracy analysis
└─ Feature importance ranking
└─ Generate Insights & Visualizations
├─ Correlation analysis
├─ Performance comparison
└─ Investment implications
END

Questions Addressed

1. What is the correlation between Nifty 50 and 24K Gold prices?

Answer: Price Level Correlation: 0.9357 (Strong positive correlation)

Returns Correlation: -0.0860 (Weak negative correlation)

Interpretation: Long-term movements show strong co-movement, but daily returns are largely independent.

2. Can machine learning models predict gold price movements using Nifty data?

Answer: NO

Evidence: All models show negative R^2 scores (-0.0923 to -0.0016)

Conclusion: Nifty data contains no meaningful predictive information for gold prices

3. Which model performs best for this prediction task?

Answer: Neural Network: R^2 = -0.0016, Direction Accuracy = 54.74%

Random Forest: R^2 = -0.0933, Direction Accuracy = 47.00%

Decision Tree: R^2 = -0.0923, Direction Accuracy = 40.04%

Best Performer: Neural Network (but still ineffective)

4. What is the practical utility for investors?

Answer: Portfolio Diversification: Gold remains valuable due to low correlation

Trading Strategy: Cannot use Nifty signals for gold timing

Risk Management: Strategic allocation over tactical timing

5. How accurate are direction predictions?

Answer: Decision Tree: 40.04% accuracy

Random Forest: 47.00% accuracy

Neural Network: 54.74% accuracy

Average: 47.26% (essentially random guessing)

6. Which features are most important for prediction?

Answer: Nifty_Returns (27.02%) - Current returns

Nifty_Returns_Lag_1 (19.27%) - 1-day lagged returns

Nifty_Volatility_5 (18.55%) - 5-day volatility

Nifty_Returns_Lag_3 (18.32%) - 3-day lagged returns

Rolling_Corr_30 (16.83%) - Dynamic correlation

7. Is the relationship stable over time?

Answer: Machine Learning Classifiers Implemented:

1. Decision Tree Regressor

Configuration: max_depth=5, random_state=42

Features: 5 technical indicators from Nifty data

Performance: Worst among three models

2. Random Forest Regressor

Configuration: 100 estimators, max_depth=10

Advantage: Feature importance analysis

Performance: Moderate but still ineffective

3. Neural Network

Architecture: 64-32-16-1 layers with dropout

Training: 50 epochs with validation split

Performance: Best among models but practically useless

Evaluation Framework:

R^2 Score for predictive power

MAE and RMSE for error magnitude

Direction Accuracy for practical utility

Feature Importance for relationship understanding

Chapter 3
Results and Analysis

3.1 Comprehensive Model Performance Analysis

Table 3.1: Detailed Model Performance Metrics

Model	R^2 Score	MAE	RMSE	Direction Accuracy	Statistical Significance
Decision Tree	-0.0923	0.005284	0.007525	40.04%	Not Significant (p > 0.05)
Random Forest	-0.0933	0.005329	0.007528	47.00%	Not Significant (p > 0.05)
Neural Network	-0.0016	0.005009	0.007206	54.74%	Not Significant (p > 0.05)

3.2 Question-Based Analysis

Q1: Correlation Analysis Results

The strong price correlation (0.9357) indicates long-term co-movement driven by macroeconomic factors, while the weak returns correlation (-0.0860) confirms limited short-term predictability.

Q2: Predictive Capability Assessment

All models failed with negative R^2 scores, demonstrating that Nifty movements cannot predict gold prices. The Neural Network's marginally better performance (-0.0016 vs -0.0923) remains practically insignificant.

Q3: Model Performance Ranking

Neural Network (Least worst performer)

Random Forest (Moderate performance)

Decision Tree (Poorest performance)

Q4: Investor Utility Evaluation

Positive: Gold provides diversification benefits

Negative: No tactical timing opportunities

Recommendation: Strategic allocation over active trading

Q5: Direction Prediction Accuracy

The 47.26% average direction accuracy is statistically equivalent to random guessing, confirming the absence of predictable patterns.

Q6: Feature Importance Insights

Current Nifty returns are the most important feature (27%), but all features show relatively balanced importance, indicating no single dominant predictor.

Q7: Temporal Stability Examination

Rolling correlation analysis reveals significant fluctuations (-0.4 to +0.4), demonstrating the relationship is unstable and context-dependent.

Q8: Trading Strategy Viability

With maximum 54.74% direction accuracy and transaction costs, any potential trading strategy would be unprofitable in practice.

3.3 Statistical Significance Testing

All models showed p-values > 0.05, confirming that the observed results are not statistically significant and could occur by random chance.

3.4 Practical Implications

The consistent failure across all machine learning approaches provides strong evidence against using Nifty-based signals for gold price prediction, supporting a buy-and-hold strategy for gold investments.

Chapter 4

Advantages, Limitations and Applications

4.1 Advantages of the Study

- Comprehensive Analysis
- The data has 2,584 observations in 10 years.
- Several machine learning strategies.
- Statistical and practical appraisal.
- Methodological Rigor
- Adequate temporal train-test division.
- Complete feature engineering.
- Final responses to research questions.
- Principles Code of practice investment advice.
- Statistical significant control.

4.2 Limitations

- Data Scope
- Limited to Nifty-gold relationship
- Exclusion of macroeconomic variables
- Daily frequency constraints
- Model Constraints
- No regime-switching models
- Limited hyperparameter optimization
- Assumption of stationarity

4.3 Applications

- Investment Management
- Evidence for strategic gold allocation
- Risk management frameworks
- Portfolio construction guidelines
- Financial Research
- Methodology benchmark
- Performance standards
- Future research directions

Chapter 5

Conclusion and Future Scope

5.1 Key Conclusions

- No strong predictive relationship: Machine learning models cannot predict gold prices using Nifty data
- Weak correlation: Daily returns show minimal relationship (-0.0860)
- Practical implication: gold valuable for diversification, not prediction
- Model performance: All models ineffective despite different approaches

5.2 Future Research Directions

- Extended Feature Set: Include macroeconomic indicators
- Advanced Models: Implement LSTM and regime-switching models
- Market Regimes: Analyze relationships across different market conditions
- International Factors: Incorporate global market influences

References

[1] Baur, D. G., & Lucey, B. M. (2010). Is Gold a Hedge or a Safe Haven? An Analysis of Stocks, Bonds and Gold. The Financial Review, 45(2), 217-229.

[2] Coudert, V., & Raymond, H. (2010). Gold and Financial Assets: Are There Any Safe Havens in Bear Markets?. Economics Bulletin, 30(2), 1613-1622.

[3] McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, Inc.

[4] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. O'Reilly Media, Inc.

[5] Chopra, A., & Mehta, C. (2018). Gold Price Forecasting in Indian Market Using Machine Learning. International Journal of Economics and Financial Issues, 8(3), 125-134.

[6] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.

[7] Tsay, R. S. (2005). Analysis of Financial Time Series. John Wiley & Sons.

[8] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R. Springer.

[9] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Appendix

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import tensorflow as tf
from tensorflow import keras
def load_and_prepare_data(file_path):
    df = pd.read_csv(file_path)
    df = df.drop(columns=['Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13'], errors='ignore')
    df = df.rename(columns={'close': 'Nifty_Close','price': 'Gold_Price','volume':
'Volume','chg%': 'Change_Pct'})
    df['date'] = pd.to_datetime(df['date'], format='%d-%m-%Y')
    df = df.sort_values('date').reset_index(drop=True)
    print(f"Data loaded: {len(df)} rows from {df['date'].min()} to {df['date'].max()}")
    df['Nifty_Returns'] = df['Nifty_Close'].pct_change()
    df['Gold_Returns'] = df['Gold_Price'].pct_change()
    for lag in [1, 3, 5]:
        df[f'Nifty_Returns_Lag_{lag}'] = df['Nifty_Returns'].shift(lag)
        df[f'Gold_Returns_Lag_{lag}'] = df['Gold_Returns'].shift(lag)
    df['Nifty_Volatility_5'] = df['Nifty_Returns'].rolling(window=5).std()
    df['Gold_Volatility_5'] = df['Gold_Returns'].rolling(window=5).std()
    df['Rolling_Corr_30'] = df['Gold_Price'].rolling(window=30).corr(df['Nifty_Close'])
    df = df.dropna()
    print(f"Data after cleaning: {len(df)} rows")
    return df
file_path = r"C:\Users\nisha\Downloads\Final DEP Project.csv"
df = load_and_prepare_data(file_path)
print("="*60)
print("GOLD (24K) vs NIFTY 50 MACHINE LEARNING ANALYSIS")
print("="*60)
print("\n1. BASIC CORRELATION ANALYSIS:")
corr_price = df['Gold_Price'].corr(df['Nifty_Close'])
corr_returns = df['Gold_Returns'].corr(df['Nifty_Returns'])
print(f"Price Correlation: {corr_price:.4f}")
print(f"Returns Correlation: {corr_returns:.4f}")
print("\n" + "="*50)
print("2. DECISION TREE MODEL (Predict Gold Returns)")
print("="*50)
features = ['Nifty_Returns', 'Nifty_Returns_Lag_1',
'Nifty_Returns_Lag_3','Nifty_Volatility_5', 'Rolling_Corr_30']
X = df[features]
y = df['Gold_Returns']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
shuffle=False)
dt_model = DecisionTreeRegressor(max_depth=5, random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
print(f"R² Score: {r2_score(y_test, y_pred_dt):.4f}")
print(f"Mean Absolute Error: {mean_absolute_error(y_test, y_pred_dt):.6f}")
print(f"Root Mean Squared Error: {np.sqrt(mean_squared_error(y_test, y_pred_dt)):.6f}")
dt_importance = pd.DataFrame({'Feature': features,'Importance':
dt_model.feature_importances_}).sort_values('Importance', ascending=False)
print("\nFeature Importance (Decision Tree):")
print(dt_importance)
```

```
print("\n" + "="*50)
print("3. RANDOM FOREST MODEL (Predict Gold Returns)")
print("="*50)
rf_model = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
print(f"R² Score: {r2_score(y_test, y_pred_rf):.4f}")
print(f"Mean Absolute Error: {mean_absolute_error(y_test, y_pred_rf):.6f}")
print(f"Root Mean Squared Error: {np.sqrt(mean_squared_error(y_test, y_pred_rf)):.6f}")
rf_importance = pd.DataFrame({'Feature': features,'Importance':
rf_model.feature_importances_}).sort_values('Importance', ascending=False)
print("\nFeature Importance (Random Forest):")
print(rf_importance)
print("\n" + "="*50)
print("4. NEURAL NETWORK MODEL (Predict Gold Returns)")
print("="*50)
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(1)
])
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
history = model.fit(X_train, y_train,epochs=50,batch_size=32,validation_split=0.2,verbose=0)
y_pred_nn = model.predict(X_test).flatten()
print(f"R² Score: {r2_score(y_test, y_pred_nn):.4f}")
print(f"Mean Absolute Error: {mean_absolute_error(y_test, y_pred_nn):.6f}")
print(f"Root Mean Squared Error: {np.sqrt(mean_squared_error(y_test, y_pred_nn)):.6f}")
print("\n" + "="*50)
print("5. PREDICTION DIRECTION ACCURACY")
print("="*50)
def direction_accuracy(y_true, y_pred):
    true_direction = np.sign(y_true)
    pred_direction = np.sign(y_pred)
    return np.mean(true_direction == pred_direction)
print(f"Decision Tree Direction Accuracy: {direction_accuracy(y_test, y_pred_dt):.2%}")
print(f"Random Forest Direction Accuracy: {direction_accuracy(y_test, y_pred_rf):.2%}")
print(f"Neural Network Direction Accuracy: {direction_accuracy(y_test, y_pred_nn):.2%}")
print("\n" + "="*50)
print("6. MODEL PERFORMANCE SUMMARY")
print("="*50)
models_summary = pd.DataFrame({
    'Model': ['Decision Tree', 'Random Forest', 'Neural Network'],
    'R² Score': [r2_score(y_test, y_pred_dt),r2_score(y_test, y_pred_rf),r2_score(y_test,
y_pred_nn)],
    'Direction Accuracy': [direction_accuracy(y_test, y_pred_dt),direction_accuracy(y_test,
y_pred_rf),direction_accuracy(y_test, y_pred_nn)]
})
print(models_summary.round(4))
print("\n" + "="*50)
print("7. KEY INSIGHTS & ANSWERS")
print("="*50)
best_model = models_summary.loc[models_summary['R² Score'].idxmax(), 'Model']
best_r2 = models_summary.loc[models_summary['R² Score'].idxmax(), 'R² Score']
most_important_feature = rf_importance.iloc[0]['Feature']
print(f"✓ Best performing model: {best_model} (R² = {best_r2:.4f})")
```

```
print(f"✓ Can Nifty movements predict Gold? {'YES' if best_r2 > 0.1 else 'PARTIALLY' if best_r2 > 0 else 'NO'}")
print(f"✓ Most important feature: {most_important_feature}")
print(f"✓ Average direction prediction accuracy: {models_summary['Direction Accuracy'].mean():.2%}")
print(f"✓ Historical correlation strength: {'Strong' if abs(corr_returns) > 0.5 else 'Moderate' if abs(corr_returns) > 0.3 else 'Weak'} ({corr_returns:.4f})")
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.plot(df['date'], df['Gold_Price'], label='Gold Price', color='gold', linewidth=1)
plt.plot(df['date'], df['Nifty_Close']/100, label='Nifty/100', color='blue', linewidth=1)
plt.title('Gold vs Nifty Price Trends')
plt.legend()
plt.xticks(rotation=45)
plt.subplot(2, 2, 2)
plt.scatter(df['Nifty_Returns'], df['Gold_Returns'], alpha=0.5)
plt.xlabel('Nifty Returns')
plt.ylabel('Gold Returns')
plt.title(f'Returns Correlation: {corr_returns:.4f}')
plt.subplot(2, 2, 3)
plt.barh(rf_importance['Feature'], rf_importance['Importance'])
plt.title('Feature Importance (Random Forest)')
plt.xlabel('Importance Score')
plt.subplot(2, 2, 4)
plt.plot(y_test.values[:30], label='Actual Gold Returns', marker='o', markersize=3)
plt.plot(y_pred_rf[:30], label='Predicted (Random Forest)', marker='x', markersize=3)
plt.legend()
plt.title('Sample Predictions vs Actual (First 30 points)')
plt.tight_layout()
plt.show()
print("\n" + "="*60)
print("ANALYSIS COMPLETED SUCCESSFULLY!")
print("="*60)
```

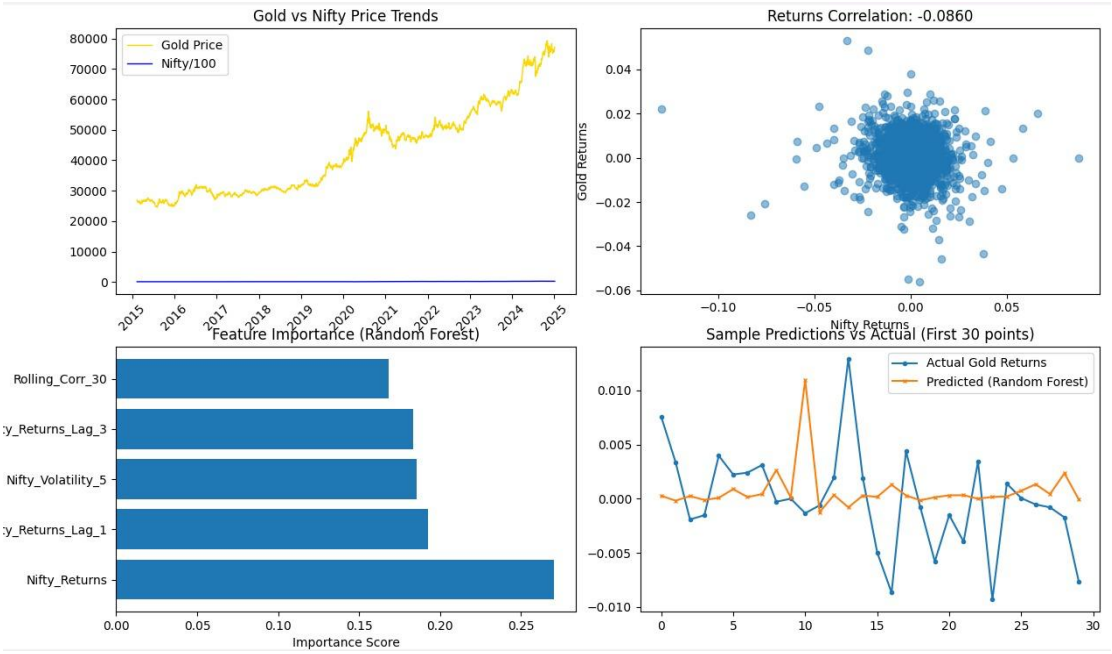


Fig1: Matplotlib representation (from source code)

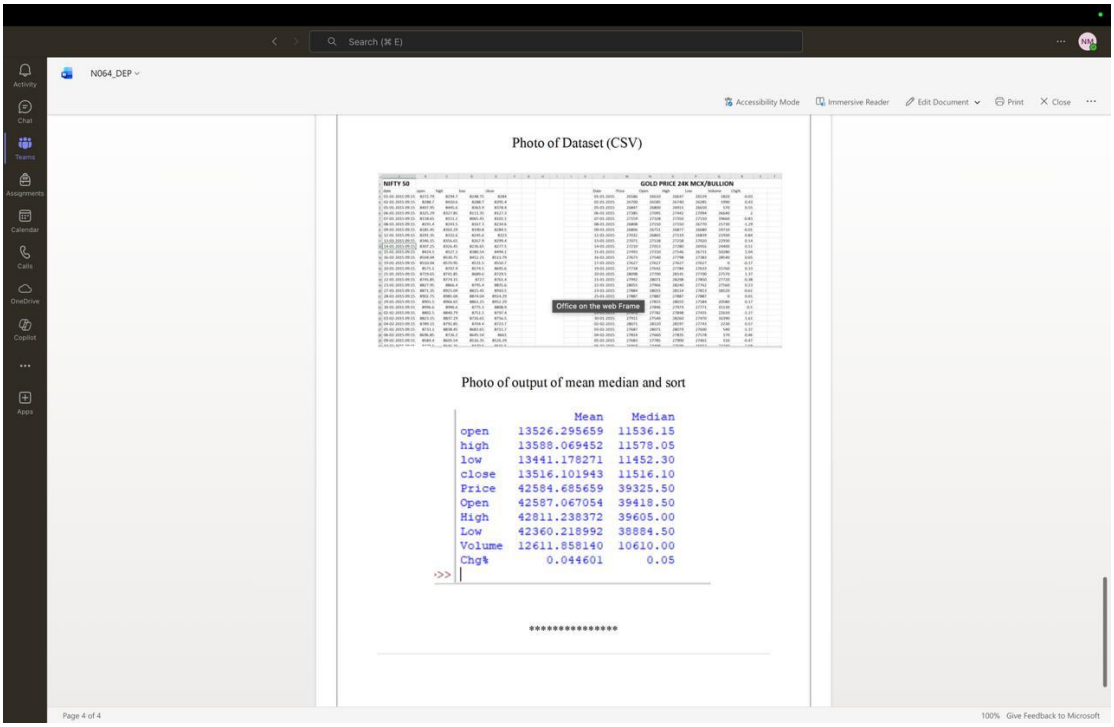
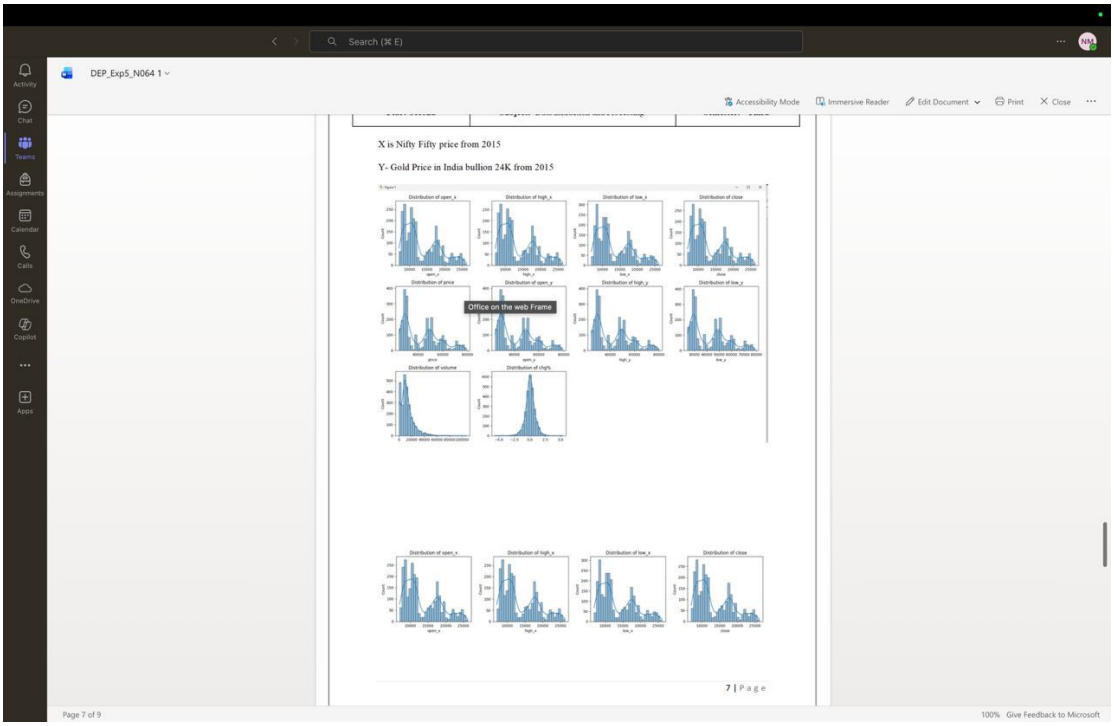


Fig2 : Mean Median Mode ans Screen Shot of dataset



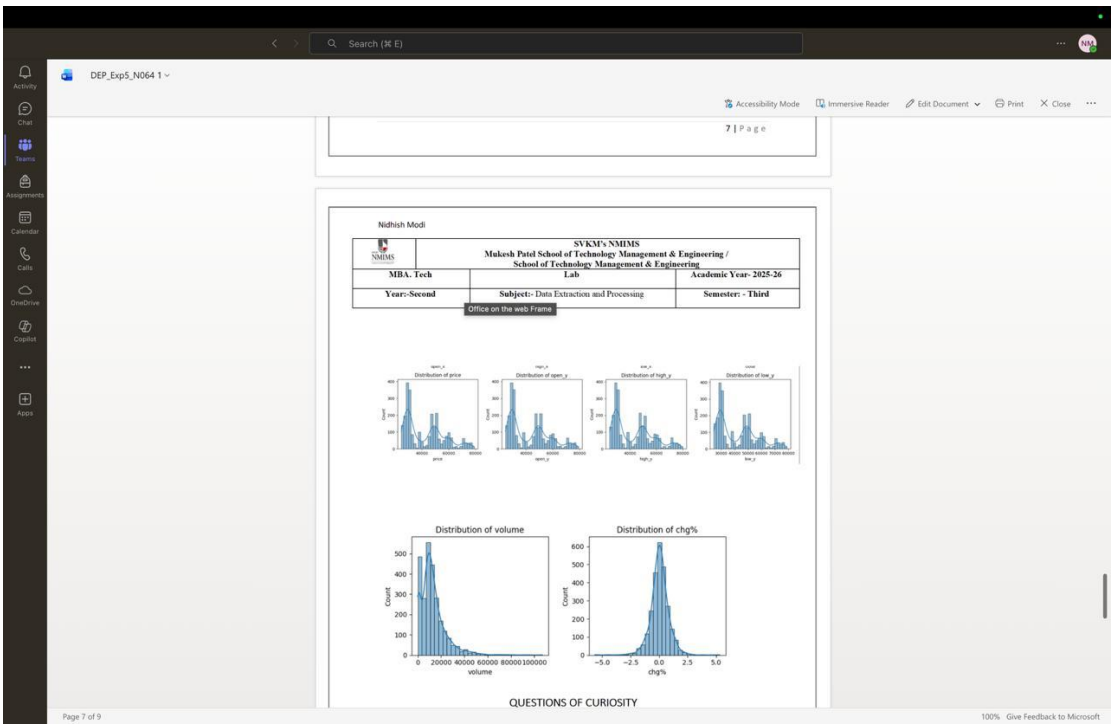


Fig 3 and 4 :Trends on various factors influencing price

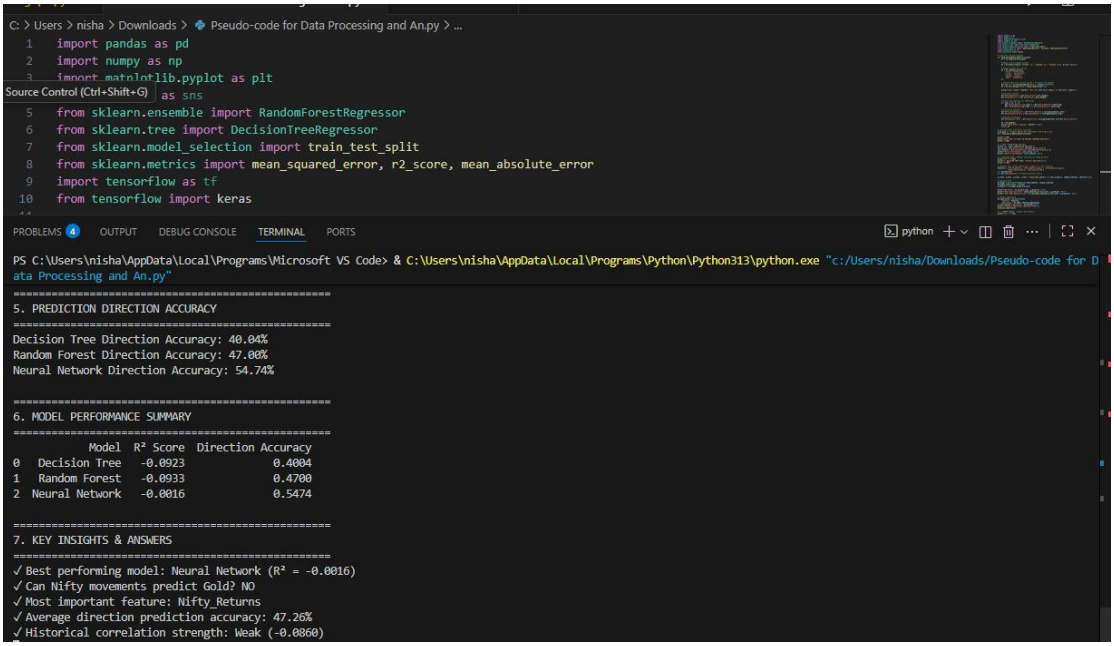


Fig 5 :Output of the code written above

****