**Quantitative Finance Portfolio**

**1. Introduction**

This portfolio highlights my practical learning journey in computational finance, where I used Python to analyze financial data, manage risk, and make investment decisions. Each project focuses on applying theoretical concepts to real-world data for better insights and outcomes.

**2. Purpose of the Portfolio**

The main purpose of this portfolio is to demonstrate how Python can be used in financial analysis, risk modeling, and investment optimization.  
It aims to:

* Apply quantitative finance techniques in practical projects.
* Showcase programming and analytical problem-solving skills.
* Bridge the gap between finance theory and real-world data application.
* Present how data-driven insights can improve financial decision-making.

**3. Portfolio Project Summaries**

**3.1: Market Risk Analysis of Tesla Stock using Historical and Monte Carlo Simulation Methods**

**Project Summary**

**Objective:**

The main objective of this project was to measure market risk of Tesla Inc.’s stock by estimating Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) using two approaches:

1. Historical Simulation Method
2. Monte Carlo Simulation Method

The goal was to understand how much an investor could potentially lose in a day under normal and extreme market conditions.

**Dataset:**

* **Source:** Investing.com
* **Period:** 1 year of Tesla’s daily stock prices
* **Data Used:** Daily closing prices, converted into daily returns for analysis

The data represents Tesla’s real market performance and served as the foundation for both simulation method.

**Methodology:**

**A. Historical Simulation Method**

* Calculated daily returns from actual Tesla stock prices.
* Sorted returns from lowest to highest.
* Determined VaR and CVaR at 90%, 95%, 99%, and 100% confidence levels.

**B. Monte Carlo Simulation Method**

* Computed mean and standard deviation of Tesla’s daily returns.
* Generated thousands of random return scenarios based on these parameters.
* Built a simulated distribution and calculated VaR and CVaR at the same confidence levels.

**Results:**

**Historical Simulation Results**

* **Mean Return:** 0.33%
* **Standard Deviation:** 4.5%
* **95% VaR:** –5.76%
* **99% VaR:** –9.63%
* **CVaR (95%):** –8.63%

*Interpretation:* Tesla shows high volatility. Past data indicates that in 95% of days, losses will not exceed 5.76%, but extreme cases can reach –9.63% or more.

**Monte Carlo Simulation Results**

* **Mean Return:** 0.10%
* **Standard Deviation:** 4.4%
* **95% VaR:** –7.08%
* **99% VaR:** –9.88%
* **CVaR (95%):** –8.82%

*Interpretation:* Monte Carlo results are smoother since they are based on probability simulations rather than actual past shocks. Risk levels are still high and similar to historical estimates.

**Comparison of Methods:**

| **Aspect** | **Historical Simulation** | **Monte Carlo Simulation** |
| --- | --- | --- |
| **Data Dependence** | Based on actual past returns | Based on simulated random returns |
| **Captures Real Market Shocks** | Yes | No (probabilistic) |
| **Flexibility for Future Scenarios** | Limited | High |
| **Result Nature** | May show extreme losses | Smoother estimates |

**Overall Conclusion:**

* Both methods indicate that Tesla stock carries significant market risk.
* In extreme market situations, Tesla’s stock could drop by around 9% or more in a single day.
* This highlights the importance of risk management for investors holding Tesla or other high-volatility stocks.

**Key Learnings:**

1. Learned how to apply Historical and Monte Carlo Simulation for risk measurement.
2. Understood that VaR estimates the maximum expected loss for a given confidence level.
3. Understood that CVaR shows the *average loss* when losses go beyond VaR.
4. Observed that Tesla’s volatility is high, making it a risky investment.
5. Realized the difference:
   * Historical method captures real past shocks,
   * Monte Carlo provides probability-based risk predictions.

**3.2. Investment Decision (Financial) Calculator using Python**

**Project Summary:**

**Introduction:**

Investment decisions are a key part of financial management. Before investing in any project, managers must evaluate whether it will generate sufficient returns to justify the cost.

This project focuses on building a Financial Calculator using Python that evaluates an investment using three important financial techniques:

1. Net Present Value (NPV) – Measures total value added by an investment considering the time value of money.
2. Internal Rate of Return (IRR) – The rate at which NPV becomes zero, showing the project’s break-even return.
3. Payback Period – The time required to recover the initial investment.

Additionally, the project includes a graphical visualization of cash flows to make the analysis more understandable and visually clear.

**Objectives:**

* To apply key finance concepts (NPV, IRR, Payback Period) through Python programming.
* To automate the investment evaluation process using Python functions.
* To visualize financial results for easier decision-making.
* To demonstrate the practical integration of financial theory and coding.

**Key Features:**

1. Net Present Value (NPV):

* If NPV > 0, the investment is profitable.
* If NPV < 0, the investment is not profitable.

2. Internal Rate of Return (IRR):

* IRR is the discount rate where NPV = 0.
* Calculated using Python’s scipy.optimize.newton method.
* Decision Rule:
  + If IRR > Discount Rate, Accept the project.
  + If IRR < Discount Rate, Reject the project.

3. Payback Period:

* Measures how many years are needed to recover the initial investment.
* Helps understand how quickly the investment’s cost is recovered.
* Limitation: It ignores the time value of money but is still a useful additional check.

4. Cash Flow Visualization:

* Created using Matplotlib.
* Red bars represent cash outflows (investment).
* Green bars represent inflows (returns).
* The chart also displays NPV, IRR, and the final decision (ACCEPT / REJECT) for better interpretation.

**Tools & Technologies Used:**

* **Python** – Main programming language.
* **Google Colab** – For code execution and implementation.
* **Matplotlib** – For cash flow visualization.
* **Scipy** – For IRR calculation using numerical optimization.

**Project Workflow:**

1. **Import Libraries** – Imported necessary packages (scipy, matplotlib).
2. **Define NPV Function** – Custom Python function to compute NPV.
3. **Define IRR Function** – Used numerical solving methods to find IRR.
4. **Input Cash Flows** – Example: [-1000, 500, 600, 700] (investment + returns).
5. **Perform Calculations** – Computed NPV, IRR, and applied decision rules.
6. **Visualization** – Displayed cash flow graph with results.
7. **Payback Period** – Calculated years required to recover investment.
8. **Final Decision** – Combined all methods to conclude **Accept or Reject.**

**Results:**

* **Input Cash Flows:** [-1000, 500, 600, 700]
* **Discount Rate:** 10%
* **Outputs:**
  + **NPV:** 272.33
  + **IRR:** 18.3%
  + **Payback Period:** 2 years
  + **Final Decision:** **ACCEPT**

**Graph Output:**

* **Year 0:** Red bar (initial investment)
* **Years 1–3:** Green bars (cash inflows)
* Graph displays **NPV, IRR, and final decision** clearly on the plot.

**Learning & Reflection:**

This project helped strengthen both financial understanding and Python programming skills.

Key learnings include:

1. Converting financial formulas like NPV, IRR, and Payback Period into working Python code.
2. Using visualization to better interpret financial results.
3. Gaining insight into real-world investment decision-making.
4. Understanding that coding can make financial analysis faster, automated, and more reliable.
5. Adding the Payback Period feature showed creativity beyond the basic scope of the project.

**3.3. Options Data Analysis for Apple Inc. (AAPL)**

**Project Summary:**

**Objective:**

To analyze AAPL options data using Python to understand market sentiment, trader behavior, and volatility patterns.

**Tools Used:**

* **Python**
* **yfinance** for data collection
* **Pandas & NumPy** for data cleaning
* **Matplotlib** for visualization

**Key Steps:**

1. **Data Collection:**
   * Downloaded AAPL stock and options data using *yfinance*.
2. **Data Cleaning:**
   * Converted key columns to numeric types and handled missing values.
3. **Put/Call Ratio (PCR):**
   * Calculated to determine market sentiment.
   * **PCR > 1 → Bearish**, **PCR < 1 → Bullish**.
4. **Top Strike Analysis:**
   * Identified top 5 strike prices with highest open interest for calls and puts.
5. **Max Pain Price:**
   * Found the strike where total option buyer losses are maximized — possible expiry target.
6. **Implied Volatility (IV) Analysis:**
   * Plotted IV vs Strike Price to observe volatility smile or skew.

**Results:**

* **PCR** revealed overall market sentiment.
* **Top strikes** showed key support and resistance levels.
* **Max Pain** identified potential expiry price zone.
* **IV curve** displayed expected market volatility trends.

**Key Learnings:**

1. How to fetch and clean real options data using Python.
2. How to calculate and interpret PCR, Max Pain, and IV.
3. How options data reflects **market psychology** and **future expectations**.

**Conclusion:**

Options analysis provides deeper insights into trader sentiment and risk expectations. Techniques like **Put/Call Ratio**, **Max Pain**, and **Implied Volatility** help understand how the market anticipates future price movements for AAPL.

**3.4. Portfolio Optimization Using Python (with Pakistani Companies)**

**Project Summary:**

**Objective:**

To find the optimal allocation of funds among five Pakistani companies — NATF, PSO, SYS, PPL, and FFC — to maximize risk-adjusted returns using the Sharpe Ratio.

**Tools Used:**

* **Python**
* **Pandas, NumPy** for data handling
* **Matplotlib** for visualization
* **SciPy (SLSQP)** for optimization

**Steps Performed:**

1. **Data Preparation:**
   * Loaded daily price data of the five companies.
   * Cleaned missing values and calculated daily returns.
2. **Statistical Analysis:**
   * Calculated mean returns and covariance matrix of the assets.
3. **Risk-Free Rate:**
   * Added a risk-free rate to calculate the Sharpe Ratio.
4. **Optimization Setup:**
   * Defined portfolio return, volatility, and Sharpe Ratio functions.
5. **Running Optimization:**
   * Used SLSQP method to maximize the Sharpe Ratio under constraints.
6. **Results (Optimal Portfolio):**
   * **Systems Limited (SYS):** ~18.06%
   * **Fauji Fertilizer Company (FFC):** ~81.93%
   * Others: ~0%
   * Calculated optimal return, volatility, and Sharpe Ratio.
7. **Efficient Frontier:**
   * Plotted efficient frontier showing best risk-return combinations.
   * Marked optimal portfolio with a red star.
8. **Visualization:**
   * Displayed efficient frontier and portfolio weights bar chart.

**Key Learnings:**

* Portfolio optimization helps achieve the best trade-off between risk and return.
* Diversification reduces risk due to stock correlations.
* The Sharpe Ratio is vital for identifying the most efficient portfolio.
* Practical constraints like no short-selling improve real-world relevance.
* FFC and SYS together formed the most optimal, high-performing portfolio.

**Conclusion:**

Using Python for portfolio optimization provided a clear, data-driven approach to asset allocation. The analysis showed that investing mainly in Fauji Fertilizer and Systems Limited yields the highest risk-adjusted return among the selected Pakistani stocks.

**3.5. Credit Risk Scoring Model Using Logistic Regression**

**Project Summary**

This project focuses on developing a Credit Risk Scoring Model using Logistic Regression to predict the likelihood that a borrower will default on a loan. Credit risk refers to the possibility that a borrower will not meet their financial obligations, and such models help financial institutions make informed lending decisions and reduce losses.

The project used Python along with libraries like Pandas, NumPy, Matplotlib, and Scikit-learn. A synthetic dataset of 1000 applicants was created, containing features such as Income, Debt, Credit Utilization, Payment History, and a derived feature Debt-to-Income Ratio. The target variable was binary — Default (1) or No Default (0).

The data was divided into 80% training and 20% testing sets. The Logistic Regression model was trained to classify borrowers based on their financial behavior. Model performance was evaluated using Confusion Matrix, Classification Report, and ROC-AUC Score.

**Results**

* **Accuracy:** 88%
* **ROC-AUC Score:** 0.88 (Excellent discrimination power)
* **Precision for Default (1):** 0.74
* **Recall for Default (1):** 0.53
* **F1-Score:** 0.62

The results show that the model performs very well in predicting non-defaulters and reasonably well for defaulters. The most important predictors of default were Credit Utilization and Debt-to-Income Ratio, while Payment History also had a notable impact. Income showed minimal influence, indicating that relative ratios are more informative than absolute income levels.

**Key Insights**

1. **High credit utilization** and **high debt-to-income ratios** significantly increase the chance of default.
2. The model’s **ROC curve** lies well above the baseline, confirming strong predictive ability.
3. **Feature engineering** (like creating Debt-to-Income Ratio) greatly improves prediction accuracy.
4. **Logistic Regression** proved effective for binary classification and interpretable in identifying key financial risk factors.

**Learnings**

Through this project, several technical and analytical insights were gained:

* Understanding how **quantitative models** assess borrower risk.
* Applying **Logistic Regression** to predict probabilities of default.
* Evaluating model performance through multiple metrics.
* Interpreting **coefficients** to understand the financial behavior influencing default risk.

**3.6. Stock Price Prediction Using ARIMA + GARCH: Tesla (TSLA) Example**

**Project Summary**

This project focuses on forecasting the next-day return, volatility, and price of Tesla (TSLA) stock using a combination of ARIMA (AutoRegressive Integrated Moving Average) and GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) models. These models together capture both the predictable trends and volatility behavior in financial time series, helping investors understand potential future movements and risk levels.

**Objective**

The main goal was to predict next-day return and volatility for Tesla stock based on its historical price data.  
Tesla was chosen because it is a highly traded and volatile stock, making it suitable for testing time-series forecasting models. Historical data was obtained from Yahoo Finance for the period January 1, 2018 – December 31, 2023.

**Methodology**

1. **Data Collection:**
   * Tesla’s adjusted closing prices were downloaded using the *yfinance* library.
   * Data was checked for missing values.
2. **Return Calculation:**
   * Daily **log-returns** were computed to ensure stationarity, which is required for ARIMA modeling.
3. **ARIMA Model:**
   * The ARIMA(1,0,1) model was used to capture the mean or predictable component of stock returns.
   * It provided the expected log-return for the next trading day.
4. **ARCH Effect Testing:**
   * The residuals from the ARIMA model were tested for volatility clustering using the ARCH test.
   * Since the p-value was < 0.05, this confirmed the presence of heteroskedasticity, justifying GARCH modeling.
5. **GARCH Model:**
   * The GARCH(1,1) model was applied to model the conditional variance (volatility).
   * It forecasted volatility for the next five trading days, showing how uncertainty changes over time.
6. **Forecasting:**
   * The ARIMA model predicted a next-day log-return of 0.00194 (≈0.19% expected increase).
   * With the last observed price of $248.48, the forecasted next-day price was $248.96.
   * The GARCH model forecasted a 1-day volatility range of approximately 2.94%, giving a price interval of $244.20 – $253.85.
7. **Diagnostics:**
   * **Ljung-Box test** confirmed that ARIMA residuals were not autocorrelated (p = 0.201), indicating a good fit.
   * However, squared residuals showed strong autocorrelation (p ≈ 1.5e-27), confirming volatility clustering — hence validating GARCH usage.

**Results:**

| **Metric** | **Value** |
| --- | --- |
| Last Observed Price | **$248.48** |
| Forecasted Next-Day Price | **$248.96** |
| Expected Return | **+0.19%** |
| 1-Day Volatility Range | **$244.20 → $253.85** |

**Interpretation:**

* The model predicts a slight upward movement in Tesla’s price.
* The volatility range reflects realistic uncertainty, showing the potential risk around the forecast.
* Investors can use this range to gauge expected variability and risk exposure.

**Conclusion**

The combination of ARIMA + GARCH provides a powerful and practical framework for stock price forecasting:

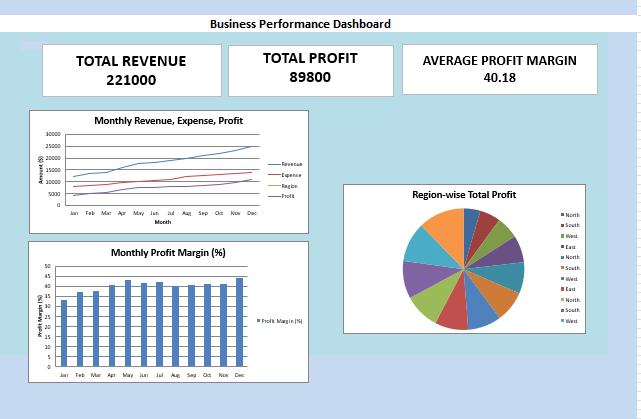
* ARIMA captures the mean (expected return).
* GARCH models the volatility (risk or uncertainty).  
  Together, they provide both point forecasts and confidence intervals, giving investors a more complete picture.

The model achieved reliable results and realistic forecasts, proving effective for educational and practical purposes.  
Possible future improvements include using EGARCH or GJR-GARCH to capture asymmetric volatility, testing heavy-tailed distributions, or extending to multi-day forecasts.

**Key Learnings**

1. ARIMA effectively models mean returns in financial data.
2. Volatility clustering exists in markets and can be captured using GARCH.
3. The ARIMA + GARCH combination provides insights into both expected price and risk range.
4. Model diagnostics (like Ljung-Box) are essential to confirm reliability.
5. Python tools make financial data analysis efficient, reproducible, and visually interpretable.
6. Understanding both return and volatility helps investors make data-driven decisions.

**3.7. Business Performance Dashboard:**

****

**Objective:**

To create an Excel-based dashboard that tracks and analyzes the company’s financial performance across the year using key business metrics and visual insights.

**Tools Used:**

* Microsoft Excel (Charts, Pivot Tables, KPIs, and Dashboard Design

**Key Metrics:**

* **Total Revenue:** 221,000
* **Total Profit:** 89,800
* **Average Profit Margin:** 40.18%

**Visual Components:**

1. **KPI Cards:** Display total revenue, total profit, and average profit margin.
2. **Monthly Revenue, Expense & Profit Chart:** Shows performance trends across months.
3. **Monthly Profit Margin (%):** Highlights profitability efficiency.
4. **Region-wise Total Profit (Pie Chart):** Shows contribution of each region to overall profit.

**Insights:**

* Consistent increase in revenue and profit over the year.
* Average profit margin remained above 40%, showing strong financial control.
* Regional analysis identifies top-performing and low-performing areas.

**Conclusion:**

The Business Performance Dashboard effectively presents a clear and concise view of financial performance. It enables management to make data-driven decisions, monitor monthly growth, and enhance strategic business planning.

**4.Extended Project Summary**

**1. Project Chosen & Why**

The project I chose for my extended work is Credit Risk Modeling.  
I selected this project because it is one of the most practical and important applications of machine learning in the finance and banking sector.  
In real life, banks need to decide whether a person is likely to repay a loan or default.  
Using data and predictive models helps them make smarter, safer decisions instead of relying only on manual judgment.

I personally enjoyed this project because it involves both financial understanding (like debt, income, and payment behavior) and technical learning (like regression, accuracy scores, and ROC curves).  
It gave me a chance to apply data analysis and machine learning concepts to a real-world problem that financial institutions face every day.

**2. Enhancements Made**

In the original version of my project, I used only one algorithm — Logistic Regression — to predict whether a person will default or not.  
While Logistic Regression worked well, I wanted to see if I could make the model more powerful and accurate.

For the extended version, I added one new model:

* **Random Forest Classifier**

**What Random Forest does:**  
It creates many small decision trees and then combines their results to make a final decision.  
Because of this, it can capture complex relationships between features such as income, debt, and payment history.  
It doesn’t assume the data is linear like Logistic Regression does — which makes it more flexible and accurate.

**Steps taken for enhancement:**

1. Trained both models — Logistic Regression and Random Forest — on the same dataset.
2. Calculated evaluation metrics like Accuracy, Precision, Recall, and ROC-AUC score for both.
3. Compared their results in a simple table and ROC Curve plot.

This enhancement was simple to implement but made the project much stronger and more practical.

**3. Results and Impact**

After adding the Random Forest model, I compared both models’ side by side.

| **Model** | **ROC-AUC Score** | **Result** |
| --- | --- | --- |
| Logistic Regression | **0.8835** | Good baseline model |
| Random Forest | **0.9982** | Excellent, near-perfect performance |

**What these results mean:**

* The ROC-AUC score tells how well a model can separate defaulters from non-defaulters.
* A score closer to 1.0 means excellent accuracy.
* My original model (Logistic Regression) had a good score of 0.88,  
  but after adding Random Forest, the score improved to 0.99 — almost perfect.

**Impact of this improvement:**

* Random Forest identified more real defaulters (better recall).
* It reduced wrong predictions, meaning fewer safe customers were marked as risky.
* This kind of improvement is extremely valuable for banks, as it can help reduce financial losses and make lending safer.

In short, adding Random Forest made the model smarter, more accurate, and more reliable.

**4. Key Takeaways**

Here’s what I learned and understood from this extended version of the project:

1. **Better Models Give Better Results:**  
   Simple models like Logistic Regression are good for understanding, but advanced models like Random Forest give better accuracy and performance.
2. **Ensemble Learning Concept:**  
   I learned how Random Forest combines multiple decision trees to improve predictions — this idea of “many models working together” is called ensemble learning.
3. **Evaluating Models:**  
   I understood how to use metrics like Confusion Matrix, Classification Report, and especially **ROC-AUC** to measure model performance.
4. **Feature Importance:**  
   Some financial features such as **Credit Utilization** and **Debt-to-Income Ratio** play a bigger role in predicting default risk.
5. **Real-World Relevance:**  
   This type of project is used in the real banking industry for **credit scoring**, **loan approval**, and **risk assessment**, so it has real practical value.
6. **Practical Learning:**  
   This project improved my understanding of Python libraries like **Pandas, scikit-learn, and Matplotlib**, and how to train, test, and visualize ML models.

**5. Conclusion**

The extended version of my Credit Risk Modeling project shows how a small improvement — adding just one new model — can make a big difference in results.  
While Logistic Regression gave good baseline predictions, Random Forest significantly improved performance, achieving a very high ROC-AUC score of 0.9982.

This experiment taught me the importance of trying multiple models and comparing them before deciding which one works best.  
Overall, this project helped me understand how machine learning can make better financial decisions, reduce risk, and support smarter lending policies in the banking sector.

**5. Skills Gained**

* Python (NumPy, Pandas, Matplotlib, Scipy, yfinance)
* Portfolio Theory & Sharpe Ratio Optimization
* Financial Data Cleaning and Preprocessing
* Data Visualization & Risk Analysis
* Application of Machine Learning in Finance
* Understanding of Market Sentiment & Volatility

**6. Key Takeaways**

Through these projects, I developed a strong understanding of:

* How data-driven finance works.
* How Python simplifies quantitative modeling.
* The importance of diversification, risk analysis, and predictive modeling.
* Using programming as a tool to enhance financial decision-making.

**7. Conclusion**

This portfolio reflects the integration of finance, statistics, and programming.  
Each project deepened my understanding of how financial models are built, tested, and applied in real markets.  
It also enhanced my analytical thinking, problem-solving, and coding skills — key strengths for a career in computational finance.