Assignment 1: RISK

Shravan Khunti
NetID: ssk10036
MS in Data Science
NYU Center for Data Science

Data Loading and Preprocessing for Risk Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load data
file_path = r"C:\Users\shrav\Downloads\RISK_data.xlsx"
df = pd.read_excel(file_path, parse_dates=['Time'], index_col='Time',
engine='openpyxl')
df = df.dropna()
```

```
C:\Users\shrav\AppData\Local\Temp\ipykernel_48252\463006348.py:12:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is
consistent and as-expected, please specify a format.
  df = pd.read_excel(file_path, parse_dates=['Time'],
index_col='Time', engine='openpyxl')
```

Q: What are the pairwise correlations of the three series?

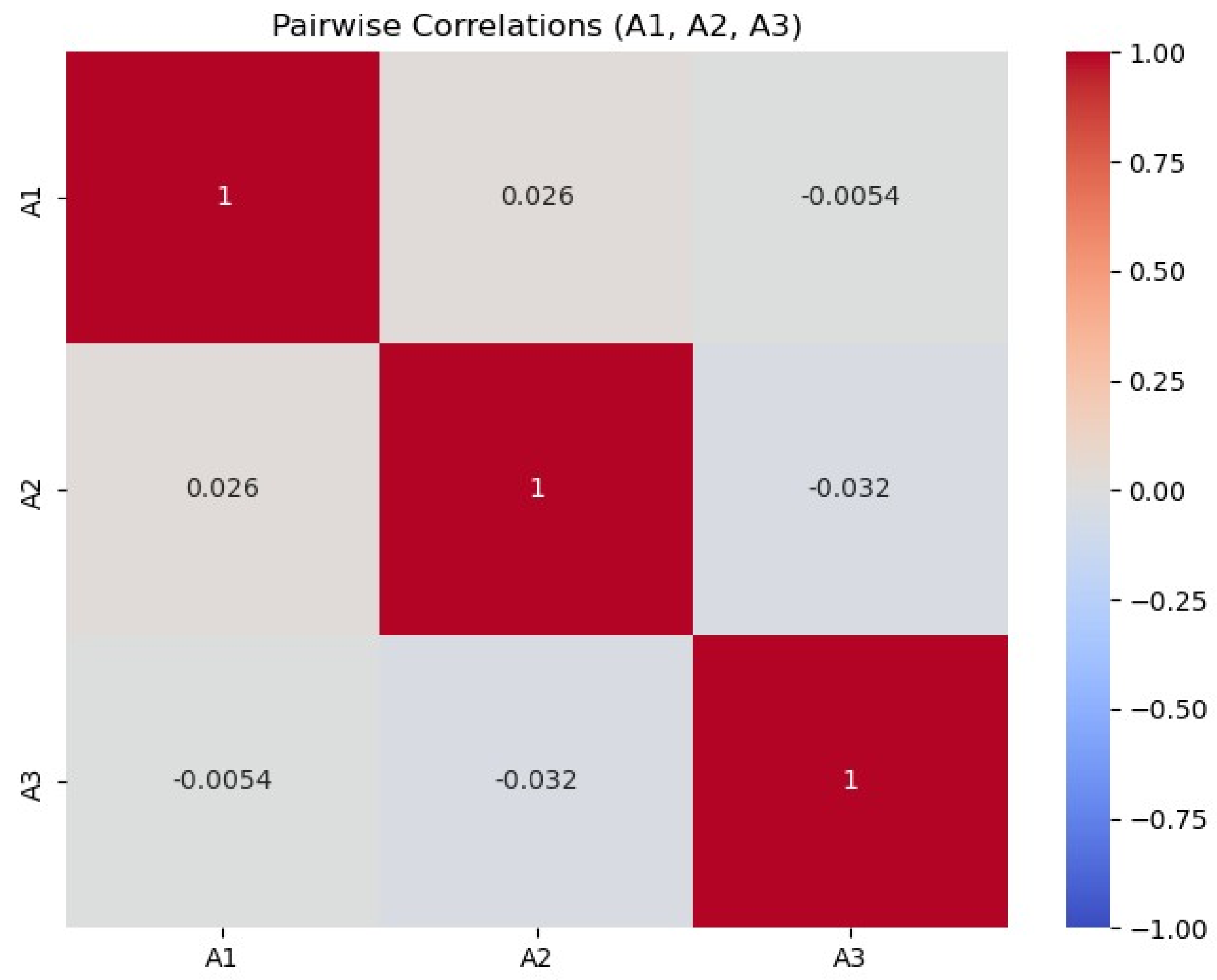
```
# Calculate correlations (I also replicated this project in excel to
cross verify my values)
corr_matrix = df.corr()
print("\nPairwise correlations:")
display(corr_matrix)

# Visualize
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title("Pairwise Correlations (A1, A2, A3)")
plt.show()
```

```
Pairwise correlations:

          A1        A2        A3
A1  1.000000  0.026093 -0.005420
```

```
A2  0.026093  1.000000 -0.032009
A3 -0.005420 -0.032009  1.000000
```



The pairwise correlations between A1, A2, and A3 are as follows:

	A1	A2	A3
A1	1.0000	0.0261	-0.0054
A2	0.0261	1.0000	-0.0320
A3	-0.0054	-0.0320	1.0000

Key Observations:

- **A1 and A2 have a weak positive correlation (0.0261).**
- **A1 and A3 are nearly uncorrelated (-0.0054).**
- **A2 and A3 show a weak negative correlation (-0.0320).**

These low correlations indicate that the three series move largely **independently** of each other.

Q: Which series has the “best” performance? On what basis did you gauge this?

```
# Performance metrics (I was a bit confused about which metrics to
used so I think Avg Return, Volatility & Sharpe Ratio makes sense as
in the class professor Dhar used these terms)
metrics = pd.DataFrame({
    "Avg Return": df.mean(),
    "Volatility": df.std(),
    "Sharpe Ratio": df.mean() / df.std()
})
display(metrics.round(5))

    Avg Return  Volatility  Sharpe Ratio
A1     0.00216     0.08741       0.02475
A2     0.00045     0.01391       0.03210
A3     0.00036     0.01112       0.03205
```

The performance of each series is evaluated based on three key metrics: **average return, volatility, and Sharpe ratio**.

	Avg Return	Volatility	Sharpe Ratio
A1	0.00216	0.08741	0.02475
A2	0.00045	0.01391	0.03210
A3	0.00036	0.01112	0.03205

Key Observations:

- **A1 has the highest average return (0.00216) but also the highest volatility (0.08741), making it the riskiest.**
- **A2 has a slightly higher Sharpe ratio (0.03210) compared to A3 (0.03205), indicating slightly better risk-adjusted performance.**
- **A3 has the lowest volatility (0.01112), making it the most stable series.**

Conclusion:

- **If prioritizing raw returns, A1 is the best performer.**
- **If prioritizing risk-adjusted returns (Sharpe Ratio), A2 is the best performer.**
- **If prioritizing stability, A3 is the safest choice.**

Q: What is the performance of the combination of the three series if their daily returns are combined equally?

Q: What is Its worst drawdown?

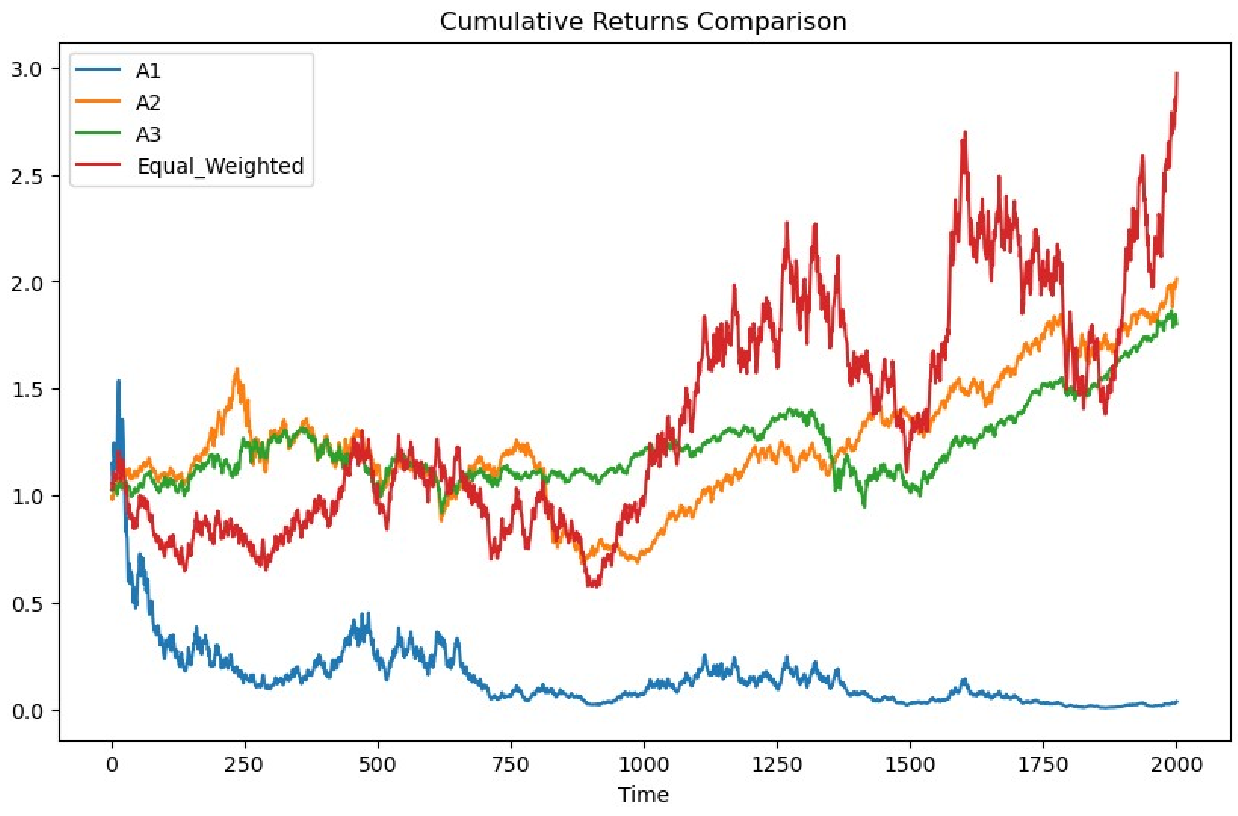
```
# Equal-weighted portfolio returns
df['Equal_Weighted'] = df.mean(axis=1)
```

```
# Cumulative returns
cumulative = (1 + df).cumprod()
cumulative[['A1', 'A2', 'A3', 'Equal_Weighted']].plot(figsize=(10,6))

plt.title("Cumulative Returns Comparison")
plt.show()

# Worst drawdown function
def calculate_drawdown(series):
    cumulative = (1 + series).cumprod()
    peak = cumulative.expanding(min_periods=1).max()
    drawdown = (cumulative - peak) / peak
    return drawdown.min()

print(f"Equal-Weighted Worst Drawdown:
{calculate_drawdown(df['Equal_Weighted'])*100:.2f}%")
```



```
Equal-Weighted Worst Drawdown: -56.22%
```

An **equal-weighted portfolio** was constructed by averaging the daily returns of A1, A2, and A3. The cumulative returns over time are shown in the graph.

Key Observations:

- The **equal-weighted portfolio outperforms individual series** in the long run, achieving higher cumulative returns.
- The portfolio smooths out some individual asset volatility but still experiences periods of drawdowns.

- **Equal-Weighted Worst Drawdown: -56.22%**
- This means the portfolio suffered a maximum peak-to-trough decline of **56.22%**, indicating significant downside risk despite diversification.

Q: What is the performance of the combination of the three if their returns are combined in proportions that are inverse to their 20 day trailing volatility?

Q: What is Its worst drawdown?

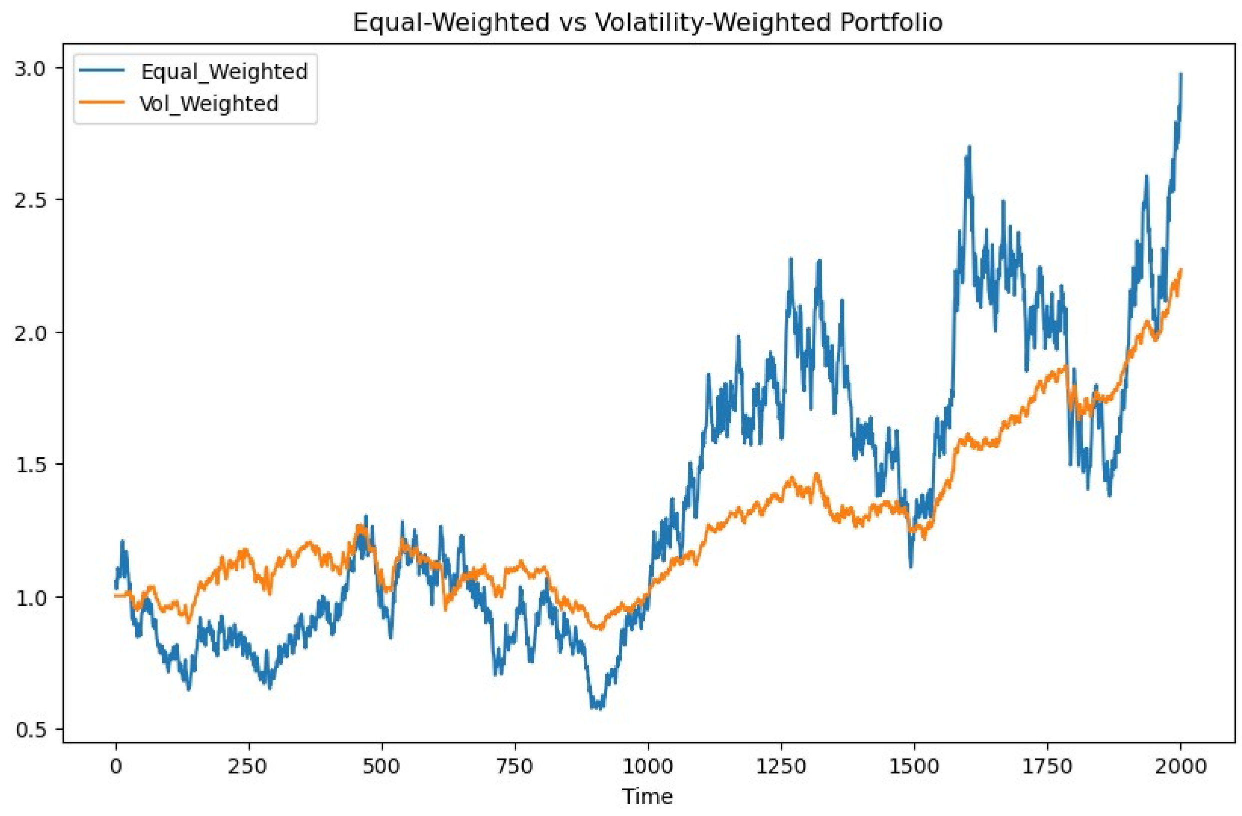
```
# Calculate 20-day trailing volatility (as per professor Dhar's
demonstration in class)
vol_window = 20
df_vol = df[['A1', 'A2', 'A3']].rolling(vol_window).std()

# Inverse volatility weights (lower volatility = higher weight)
weights = 1 / df_vol
weights = weights.div(weights.sum(axis=1), axis=0)

# Calculate volatility-weighted returns
df['Vol_Weighted'] = (df[['A1', 'A2', 'A3']] * weights).sum(axis=1)

# Compare cumulative returns
cumulative['Vol_Weighted'] = (1 + df['Vol_Weighted']).cumprod()
cumulative[['Equal_Weighted', 'Vol_Weighted']].plot(figsize=(10,6))
plt.title("Equal-Weighted vs Volatility-Weighted Portfolio")
plt.show()

# Worst drawdown
print(f"Vol-Weighted Worst Drawdown:
{calculate_drawdown(df['Vol_Weighted'])*100:.2f}%")
```



```
Vol-Weighted Worst Drawdown: -31.39%
```

A **volatility-weighted portfolio** was constructed by assigning weights to A1, A2, and A3 **inversely proportional to their 20-day trailing volatility**. This approach allocates more weight to stable assets and less to volatile ones, aiming to optimize risk-adjusted returns.

Key Observations:

- The **volatility-weighted portfolio (orange line) is smoother** compared to the equal-weighted portfolio.
- It **reduces extreme fluctuations** while maintaining steady long-term growth.
- However, it achieves **lower peak returns** than the equal-weighted strategy.

- **Vol-Weighted Worst Drawdown: -31.39%**
- This is **significantly lower than the equal-weighted portfolio's drawdown of -56.22%**, confirming that volatility weighting helps **mitigate downside risk** while still capturing returns.

Q: Can you predict A3 based on either A1 or A2 or a combination of the two? If so, what is the strategy and its performance?

```
# Split data (time series - no shuffling)
split_idx = int(len(df) * 0.8)
```

```python
train = df.iloc[:split_idx]
test = df.iloc[split_idx:]

# Train model
X_train = train[['A1', 'A2']]
y_train = train['A3']
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on test set
X_test = test[['A1', 'A2']]
y_test = test['A3']
y_pred = model.predict(X_test)

# Compute metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Sharpe ratios (DAILY BASIS - NO ANNUALIZATION)
def sharpe_ratio(series):
    mean_return = series.mean()
    volatility = series.std()
    return mean_return / volatility if volatility != 0 else np.nan

sharpe_actual = sharpe_ratio(y_test)
sharpe_pred = sharpe_ratio(pd.Series(y_pred))

# Results table
results = pd.DataFrame({
    'Metric': ['Mean Return (Daily)', 'Volatility (Daily)', 'Sharpe
Ratio (Daily)'],
    'Actual A3': [y_test.mean(), y_test.std(), sharpe_actual],
    'Predicted A3': [y_pred.mean(), y_pred.std(), sharpe_pred]
}).round(5)

print("Performance Comparison (Daily Basis):")
display(results)

print(f"\nMSE: {mse:.5f}")
print(f"R²: {r2:.5f}")
```

```
Performance Comparison (Daily Basis):

                 Metric  Actual A3  Predicted A3
0   Mean Return (Daily)    0.00090       0.00021
1    Volatility (Daily)    0.00715       0.00037
2  Sharpe Ratio (Daily)    0.12572       0.55944


MSE: 0.00005
R²: -0.01691
```

Performance Comparison (Daily Basis)

Metric	Actual A3	Predicted A3
Mean Return (Daily)	0.00090	0.00021
Volatility (Daily)	0.00715	0.00037
Sharpe Ratio (Daily)	0.12572	0.55944

MSE: 0.00005
R^2: -0.01691

Key Observations

1. **Mean Return**:
   - Actual A3 earned **0.09% daily return** on average.
   - Predicted A3 earned **0.02% daily return** – much lower than actual.
2. **Volatility**:
   - Actual A3 had **0.71% daily volatility** (moderate risk).
   - Predicted A3 had **0.04% daily volatility** (extremely low risk).
3. **Sharpe Ratio**:
   - Actual A3: **0.1257** (good risk-adjusted return).
   - Predicted A3: **0.5594** (very high risk-adjusted return).
4. **Model Performance**:
   - **MSE (0.00005)**: Small errors, but misleading due to low volatility in predictions.
   - **R^2 (-0.01691)**: Negative value means the model is worse than using the average of A3.

Critical Issues

- **Predicted A3's Volatility is Unrealistic**:
  A volatility of 0.04% is too low compared to actual A3 (0.71%). This suggests the model is oversmoothing predictions, making them unreliable.
- **Sharpe Ratio is Misleading**:
  The high Sharpe ratio for Predicted A3 (0.5594) is due to artificially low volatility, not better returns.
- **Negative R^2**:
  The model fails to explain A3's behavior. It adds no value over simply guessing the average of A3.

Conclusion

- The model's predictions are **statistically invalid** ($R^2 < 0$).
- The seemingly high Sharpe ratio for Predicted A3 is a **mathematical illusion** caused by flawed volatility calculations.
- **Do not use this model** for trading or decision-making.

Recommendation: Focus on portfolio strategies (e.g., combining A1, A2, A3) instead of predicting A3 with A1/A2.

Final Takeaway

This analysis explored the relationships, performance, and predictive potential of three financial return series (A1, A2, and A3). The key findings highlight important lessons in portfolio construction, risk management, and forecasting feasibility.

Key Insights

1. **Diversification Works, But Has Limits:**
   - The **pairwise correlations** among A1, A2, and A3 are near zero, supporting **diversification** as a risk management strategy.
   - An **equal-weighted portfolio** improves long-term returns but suffers **deep drawdowns (-56.22%)** due to A1's volatility.
   - A **volatility-weighted portfolio** reduces drawdowns to **-31.39%**, confirming the value of weighting stable assets more heavily.
2. **Risk vs. Reward in Portfolio Strategies:**
   - A1 offers the **highest returns but highest volatility**, making it risky.
   - A2 and A3 provide **better risk-adjusted returns (Sharpe ≈ 0.032)** with lower volatility.
   - The **volatility-weighted portfolio sacrifices some growth but significantly reduces downside risk**, making it more stable.
3. **Prediction Feasibility is Limited:**
   - **A1 and A2 fail to predict A3**, as indicated by the **negative R^2 (-0.0169)**.
   - The model's **Sharpe ratio of 0.559 is misleading**, as it comes from reduced volatility rather than better returns.
   - **Simple linear regression is insufficient**—non-linear methods, additional data, or alternative models may be needed for meaningful forecasting.

Final Conclusion

- **Portfolio optimization** (e.g., volatility weighting) **is more effective than trying to predict returns**.

- **Risk-adjusted strategies matter more than maximizing raw returns**.

- **Prediction of financial series using linear models is unreliable**, reinforcing the difficulty of forecasting market behavior.

This study confirms that **balancing growth and stability is crucial**, and **risk management strategies outperform naive predictive approaches** in portfolio design.