Flutter™

Financial Markets & Big Data

Prof: Dr Niamh Wylie

Candidate: Brandon Costello 23352872

Due Date: 07/03/2025

Word Count: 3000 Max

Final Assignment

Table of Contents

• Executive Summary

Long `Flutter Entertainment (FLTR) as yield a yield curve dropping beneficiary, this is a top down macro play incorporating modern machine learning techniques learned throughout the course.

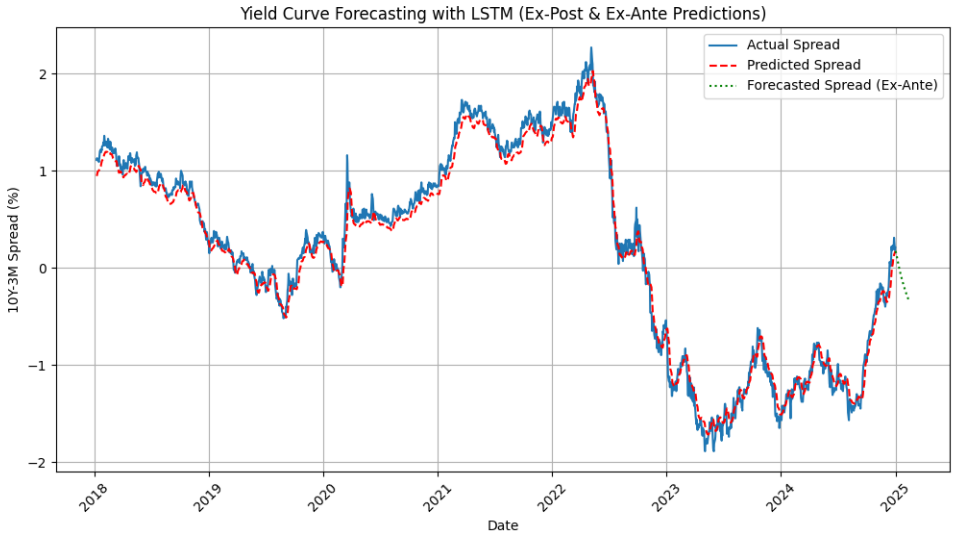
U.S rate cuts, increased liquidity and large amounts of institutional capital rotating into mid to low -beta globally diversified consumer plays. With FanDuel U.S expansion driving growth and European operations providing cash flow-stability. Flutter is positioned to absorb fed-induced liquidity flows. Whilst remaining partially insulated from U.S geopolitical uncertainty.

A dropping yield curve tends to unlock discretionary spending upside as the fed steps in to cut rates to fuel the economy.

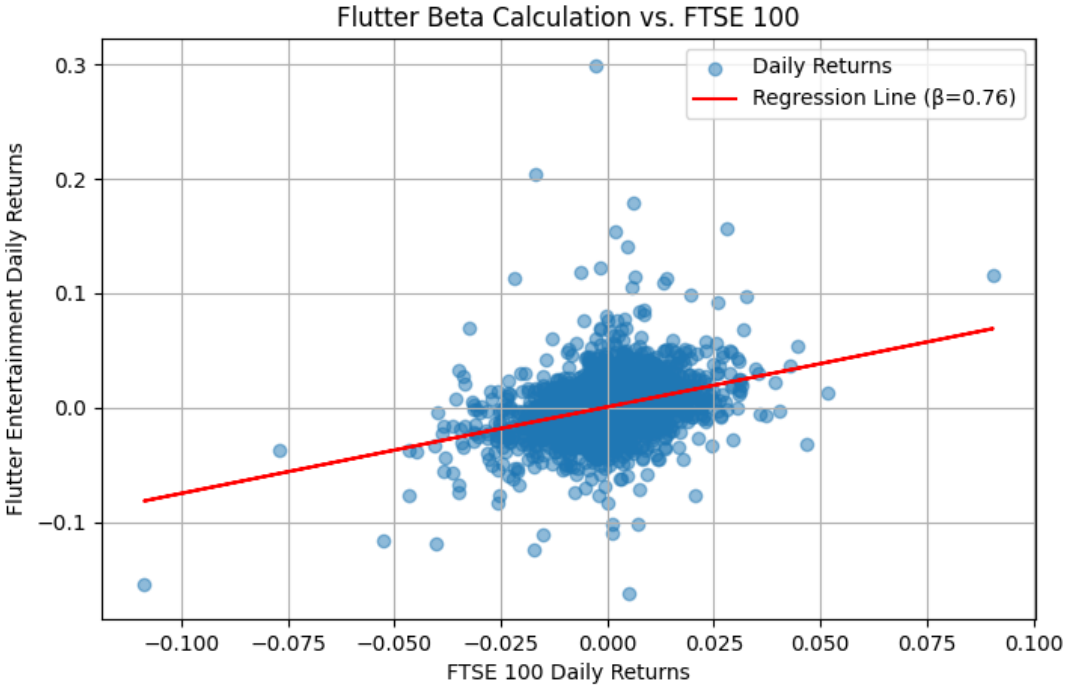
Investment Objectives

- Assess Flutter's risk-adjusted expected return using the Capital Asset Pricing Model (CAPM).
- Forecast future price movements using random forest regression and long short-term memory (LSTM) neural networks learned on the course.
- Quantify corporate stability and credit risk using Altman Z-Score
- Incorporate alternative data insights, such as options market implied volatility.
- Single Stock Purchase, with the goal of generating a single return using various models, formulae learned on the course.

Market and Economic Outlook



The yield curve is an important signal for financial markets, and my Ex-ante LSTM forecast suggest it is starting to drop sharply (green line). This means that the Federal Reserve's policies, including liquidity injections, may begin to ease tight credit conditions to circumvent recessionary fears. While geopolitical risks remain, the overall economic environment seems to support a shift in investment toward assets that can benefit from increased capital flows If the fed reduces rates even further. (DWS, 2024)



Beta

Flutter Entertainment's 0.79 beta vs. FTSE 100 suggests that its volatility is 79% of the broader UK market, making it a moderately sensitive asset in European and UK economic cycles. However, with 40% of revenue now from US operations, its S&P 500 beta of 0.50 reflects a more defensive stance against US macro factors. Using a weighted approach:

$$\beta = (0.60 \times 0.79) + (0.40 \times 0.50) = 0.67$$

CAPM	4.04%
BETA	0.67

Investment Selection & Analysis

Flutter Entertainment PLC

Flutter Entertainment plc, based in Dublin, Ireland, is a leading global company in sports betting and online gaming. It owns well-known brands like Paddy Power, Betfair, FanDuel, Sky Bet, and PokerStars, giving it a strong foothold in major markets such as the UK, Ireland, the US, Australia, and beyond.

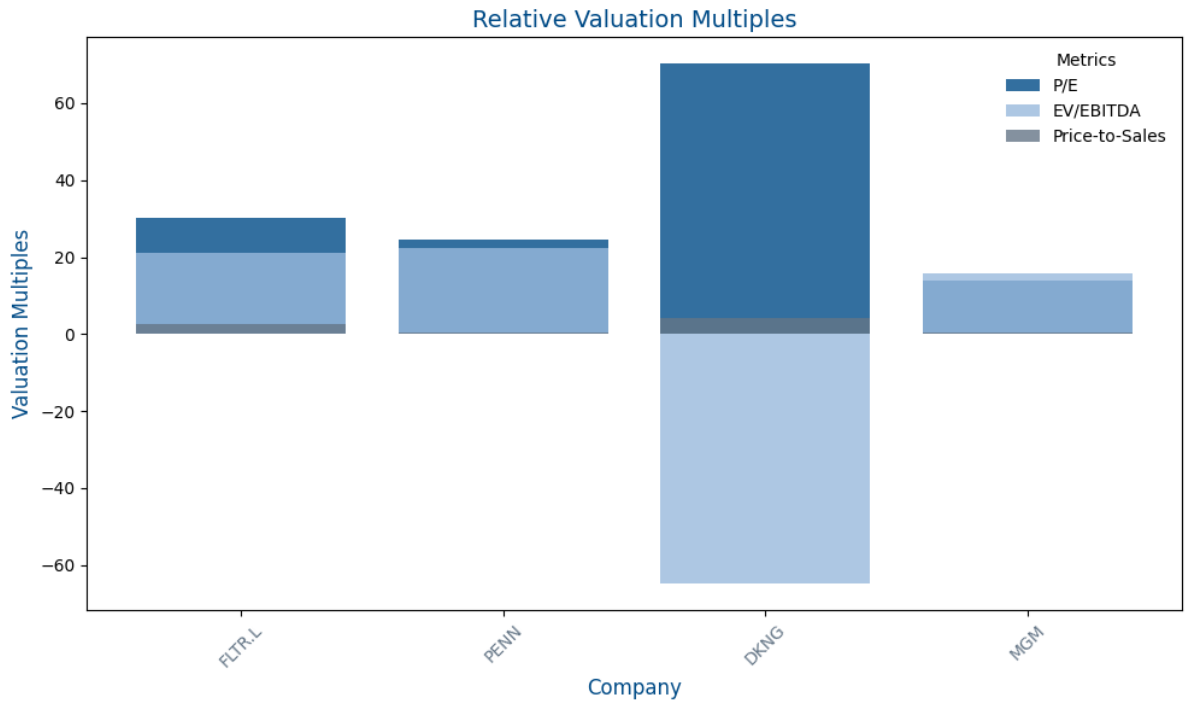
The company's success is built on high-margin online gaming, digital betting, and entertainment, with a heavy focus on technology, data analytics, and customer engagement. One of its biggest growth areas is the US, where FanDuel has become a dominant force in online sports betting, benefiting from the loosening of gambling regulations.

With a diverse range of brands and a strong competitive position, Flutter is a standout choice for this assignment. (Flutter, 2024)

1. Traditional Asset Pricing

The CAPM-implied price of €265.98, based on a beta of 0.67, supports Flutter's classification as a mid-beta consumer discretionary stock. This suggests the company offers a balance between stability and growth, especially as market liquidity conditions change. With the LSTM model predicting a steepening yield curve and the Fed expected to ease policy, the CAPM valuation provides a useful benchmark for where Flutter's stock should trade under normal market conditions.

However, CAPM does not fully account for shifts in liquidity. As bond yields fall and interest rate expectations change, institutional investors are likely to move capital into mid-beta discretionary stocks like Flutter. This could push its stock price higher than what traditional models suggest, showing how market trends can influence valuation beyond simple risk-return calculations.



Middle Ground

From a macro perspective, Flutter's valuation and financial metrics appear to align with a lower-risk thesis when compared to more speculative names like DraftKings. Its relatively moderate P/E and EV/EBITDA ratios suggest that investors are pricing it as a stable, growing company rather than a high-risk, high-reward play. This makes Flutter less risky than firms that rely heavily on future growth without strong current earnings (like DraftKings), while still offering exposure to growth trends in online gaming and sports betting.

2. Machine Learning & Predictive Analytics

While CAPM provides a risk-adjusted valuation of $265.98 (over three months with β=0.67), it is a static model that does not account for real-time market flows or liquidity-driven price adjustments. Given the LSTM forecast of a steepening yield curve and shifts in S&P and 10-year yields, the broader economic environment supports increased investment in mid-beta discretionary stocks like Flutter. Institutional investors seeking yield expansion and financial stability are likely to rotate capital into these sectors. However, since CAPM does not factor in these liquidity shifts, a more dynamic approach is needed, such as Random Forest regression.

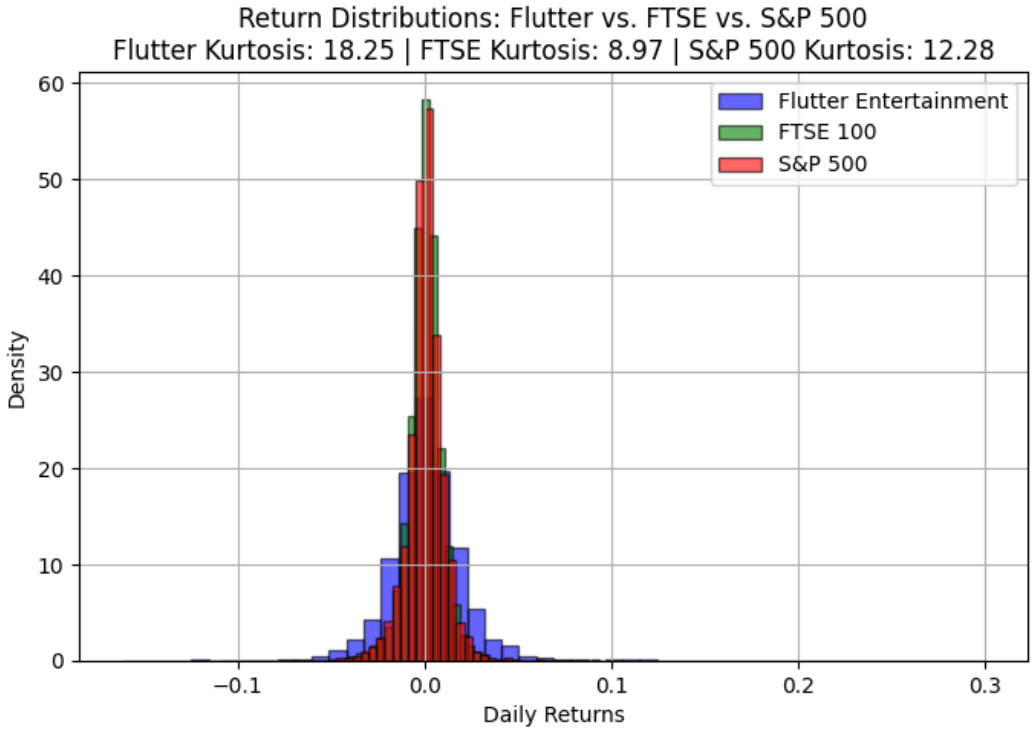
Unlike CAPM, which assumes a linear relationship between risk and return, Random Forest is a machine learning model that captures complex, non-linear interactions in market data. It works by building multiple decision trees and averaging their predictions, making it more flexible in responding to liquidity-driven changes. The Random Forest model projects an expected return of 8.08% for Flutter over the next three months, leading to an implied price of $279.34. This deviation from CAPM highlights the impact of shifting

liquidity conditions and increased discretionary spending. A feature importance analysis further supports this, showing that market returns (40.06%) and stock-specific price movements (60%) drive price action, while systematic beta (0.00%) plays no role.

As the yield curve drops and expectations of rate cuts accelerate, institutional investors are likely to increase exposure to mid-to-low beta discretionary stocks. This reinforces Flutter's position as a prime beneficiary of capital rotation. And they can also take advantage of that low rate window to fuel further expansion.

3. Risk & Volatility Analysis

Skewness & Kurtosis

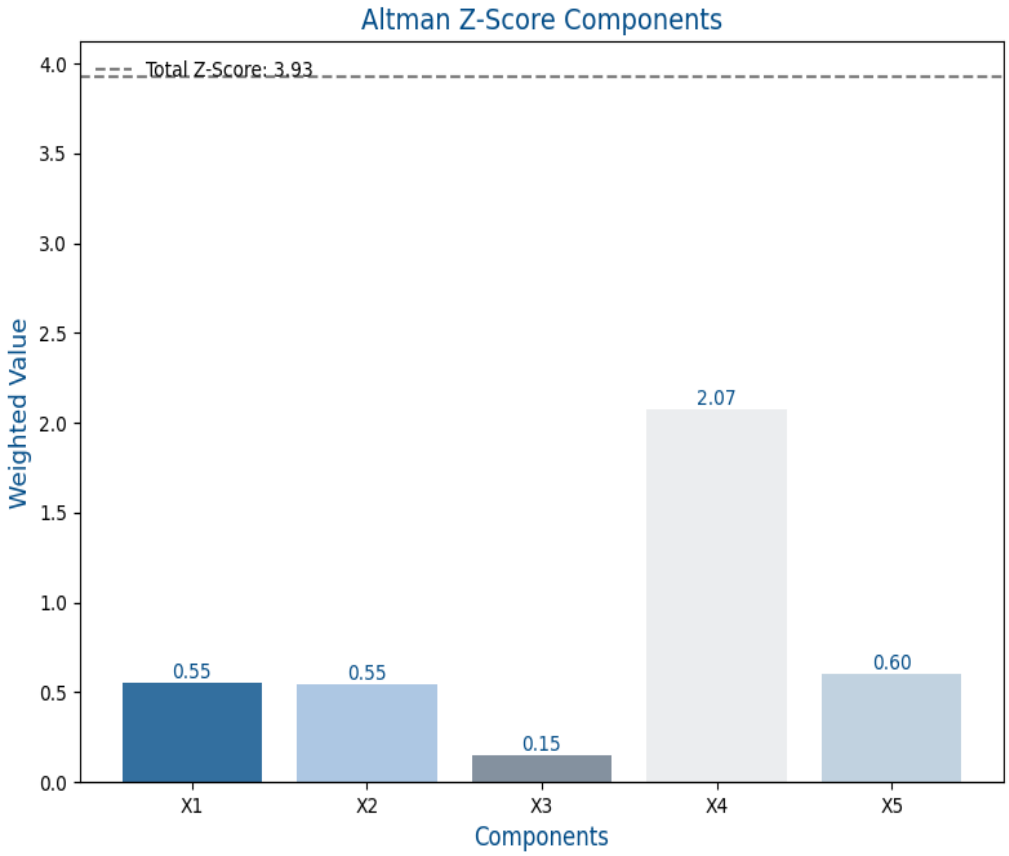


Macro	Impact on Kurtosis	Effects on FLTR
Monetary Policy	Fed and BoE Hikes/cuts Create volatility in equities	High Debt exposure makes FLTR sensitive
Regulatory Uncertainty	Rule Changes in Gambling Laws	U.K reforms / US Approvals could create massive swings in price
Retail	Retail traders amplify prices	FanDuel in the US creates a more speculative environment
Geopolitics (Trump)	War, Sanctions, Oil	Economic Uncertainty

Goldman Sachs (2024) points out that policy uncertainty plays a major role in shaping market trends. This is especially true for industries like online gambling, which are more affected by regulatory changes than other consumer discretionary sectors. This supports the idea that Flutter's risks are not random but are part of the current investment landscape, where macroeconomic factors and government policies influence stock prices more than traditional risk measures.

A clear example of this is how markets react to political decisions, such as the Trump administration's trade policies. With tariffs being placed on Canada, Mexico, Europe, and China, uncertainty rises, causing shifts in capital flows and increased market volatility. This reinforces the view that companies like Flutter are directly impacted by changes in government policy, making their stock movements less predictable

Altman Z-Score



Equation for Altman's Z-Score Model (1968):

$$Z = 1.2X_1 + 1.4X_2 + 3.3X_3 + 0.6X_4 + 1X_5$$

X_1 = Working Capital / Total Assets

X_2 = Retained Earnings / Total Assets

X_3 = Earnings Before Interest & Tax (EBIT) / Total Assets

X_4 = Market Capitalisation / Total Liabilities

X_5 = Sales / Total Assets

The chart quantitatively dissects the Altman Z-Score components, illustrating their relative contributions to a calculated value of 3.93. Each bar corresponds to a weighted input factor, offering a visual breakdown of the risk profile determinants. The score's position above the 2.99 threshold reflects a robust credit health metric, underscoring a low-risk categorization.(FAJASY, 2024)

Options sentiment

Black-Scholes Model

$$d_1 = \frac{\ln \frac{S_0}{K} + \left(r - q + \frac{\sigma^2}{2}\right) t}{\sigma \sqrt{t}}$$

Inputs into the model in python

S = 245

K = 180

T = 30/365

R = 60.91/100

Output

Theoretical price 66.65 / Market price =. 66.47 Mispricing = -0.18

implied Vol: 60.91

Expected price in one month 202.65 to 288.41 Supporting my 3 month upside move to 279.

High implied volatility means there is some potential for large moves in the market here. The 1-month expected range of 201 to 289 places my random forest price prediction of 279 well within reach. Institutional positioning suggests hedging for upside, there could be a potential price increase if implied volatility comes down.

My overall thesis depends on macro conditions remaining good, institutions rotating capital amid the steep drop in the yield curve inversion and options market staying consistent with implied volatility. With the 0.18 price difference options traders aren't likely to try and capture these inefficiencies.

Recommendation & Conclusion

Flutter Entertainment's outlook comes into focus through multiple layers of analysis. At the macro level, a drop in the yield curve could ignite looser monetary policy and set the stage for a stronger consumer discretionary sector. A weighted beta of 0.67 proves a balance between European cash flow stability and high-growth U.S. expansion via FanDuel.

Financial metrics reinforce this positioning. An Altman Z-Score of 3.93 indicates low financial risk, while CAPM suggests an expected return of 4.04%. Machine learning models, including Random Forest and LSTM, highlight how liquidity shifts and institutional positioning could drive repricing. Options data points to limited downside risk, with a likely trading range of 201 to 288, making 279 a reasonable upside target.

Buy	Current Price 7/03/25: $247
Sell	$279
Return/Upside Apr 14th.	12.9%

1-3 month time-Horizon

Narrative:

➡ ***Yield Curve drops (Trump/Tariffs) Recession fears, sticky inflation, global bond demand shift***

➡ ***Slow growth, corporate earnings decline, risk-off sentiment, failing confidence in the consumer market***

➡ ***Fed cuts to prevent capital outflows, capital rotations into equities emerge, improved consumer sentiment***

➡ ***Lower rates = capital rotation***

➡ ***Flutter share price moves with the market and Yields, 279 Price Target. As they can take advantage of that low rate window to further fuel FanDuel expansion.***

Short	Defensive sectors, Banks/defensive sectors

Personal Reflection

This assignment pushed me to think critically and engage with complex concepts in finance. In ways I hadn't before. It was so challenging but incredibly fun. I was way outside my comfort zone trying to look at monetary policy, rate cuts, applying models I learned throughout the course. Spending time buried in yield curve inversions/narratives, capital relocation. And federal reserve policy was fascinating! I could take what I was learning in the module and apply them to a stock of my choice and try to develop a macro thesis that forced me to think in multiple levels. Incredibly grateful I chose this module.

References, Figures, Tables & Source Code.

Yield Curve Forecasting

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error

file_path = "/Users/brandongwp/desktop/daily-treasury-rates.csv"
data = pd.read_csv(file_path)


data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)

1990 to 2024
data = data[(data.index >= "1990-01-01") & (data.index <= "2024-12-31")]


if '10Y-3M Spread' not in data.columns:
    if '10 Yr' in data.columns and '3 Mo' in data.columns:
        data['10Y-3M Spread'] = data['10 Yr'] - data['3 Mo']
    else:
        raise ValueError("Required columns '10 Yr' and '3 Mo' are missing.")

data.dropna(subset=['10Y-3M Spread'], inplace=True)
data = data.sort_index()

# Scale
scaler = MinMaxScaler()
yield_curve_scaled = scaler.fit_transform(data[['10Y-3M Spread']])


n_steps = 30
X, y, dates = [], [], []
```

```
if len(yield_curve_scaled) > n_steps:
    for i in range(len(yield_curve_scaled) - n_steps):
        X.append(yield_curve_scaled[i:i + n_steps])
        y.append(yield_curve_scaled[i + n_steps])
        dates.append(data.index[i + n_steps])
else:
    raise ValueError("Not enough data for LSTM sequence generation.")

X, y = np.array(X), np.array(y)
X = X.reshape((X.shape[0], X.shape[1], 1))


split_index = int(len(X) * 0.8)
X_train, X_test = X[:split_index], X[split_index:]
y_train, y_test = y[:split_index], y[split_index:]
test_dates = dates[split_index:]

# Build LSTM model
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(n_steps, 1)),
    LSTM(50, return_sequences=False),
    Dense(25, activation='relu'),
    Dense(1)
])

# Compile
model.compile(optimizer='adam', loss='mean_squared_error')

# Train model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test,
y_test))

# Ex-Post Predictions
y_pred = model.predict(X_test)

# Rescale predictions to original values
y_pred_rescaled = scaler.inverse_transform(y_pred.reshape(-1, 1))
y_test_rescaled = scaler.inverse_transform(y_test.reshape(-1, 1))

# Compute Ex-Post Model Performance Metrics
rmse = np.sqrt(mean_squared_error(y_test_rescaled, y_pred_rescaled))
mae = mean_absolute_error(y_test_rescaled, y_pred_rescaled)

print(f" Ex-Post Model Evaluation:")
print(f"RMSE: {rmse:.4f}")
print(f"MAE: {mae:.4f}")

# ------------------ Ex-Ante Forecasting: Predict Future Until April 14, 2024 -----
-------------
```

```
forecast_horizon = pd.date_range(start=test_dates[-1], periods=45, freq='D')  # Up
to April 14, 2024
future_predictions = []


future_input = X_test[-1].reshape(1, n_steps, 1)


for _ in range(len(forecast_horizon)):
    future_pred = model.predict(future_input)[0, 0]
    future_predictions.append(future_pred)


    future_input = np.roll(future_input, shift=-1, axis=1)
    future_input[0, -1, 0] = future_pred


future_predictions_rescaled =
scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))


plt.figure(figsize=(12, 6))


plt.plot(test_dates, y_test_rescaled, label="Actual Spread")
plt.plot(test_dates, y_pred_rescaled, label="Predicted Spread", linestyle="dashed",
color='red')


plt.plot(forecast_horizon, future_predictions_rescaled, label="Forecasted Spread
(Ex-Ante)", linestyle="dotted", color='green')

plt.legend()
plt.xlabel("Date")
plt.ylabel("10Y-3M Spread (%)")
plt.title("Yield Curve Forecasting with LSTM (Ex-Post & Ex-Ante Predictions)")
plt.grid()
plt.xticks(rotation=45)
plt.show()
```

Beta/CAPM

```
import pandas as pd
import numpy as np
import yfinance as yf
import statsmodels.api as sm
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Fetch historical data
flutter_ticker = "FLTR.L"  Flutter Entertainment on LSE
market_ticker = "^SPX"  # S&P 500

start_date = "2010-01-01"
end_date = "2025-01-01"

flutter_data = yf.download(flutter_ticker, start=start_date, end=end_date)
market_data = yf.download(market_ticker, start=start_date, end=end_date)

Calculate daily returns
flutter_data['Returns'] = flutter_data['Close'].pct_change()
market_data['Market_Returns'] = market_data['Close'].pct_change()

Drop NaN values
capm_data = pd.merge(flutter_data[['Returns']], market_data[['Market_Returns']],
                     left_index=True, right_index=True).dropna()

Regression to Get Beta
X = sm.add_constant(capm_data["Market_Returns"])  # Add intercept
y = capm_data["Returns"]
capm_model = sm.OLS(y, X).fit()  # Run Ordinary Least Squares Regression

 Beta & Alpha
alpha, beta = capm_model.params['const'], capm_model.params["Market_Returns"]

print(f"CAPM Beta Calculation for Flutter Entertainment (FLTR.L)")
print(f" Beta (β) vs S&P 500: {beta:.4f}")

# Step 4: CAPM Expected Return Calculation
#Fetch US 10-Year Treasury Yield (^TNX) for the Risk-Free Rate
risk_free_data = yf.download("^TNX", start="2024-01-01", end="2025-01-01")

#
latest_risk_free_rate = float(risk_free_data["Close"].iloc[-1]) / 100  # Convert
from percentage to decimal

# Adjust Market Risk Premium (Assume 6-8% for S&P 500)
market_return = capm_data['Market_Returns'].mean() * 252  # Annualized Market
Return

# Calculate Expected Return using CAPM Formula:
# Expected Return = Risk-Free Rate + Beta * (Market Return - Risk-Free Rate)
expected_return = latest_risk_free_rate + beta * (market_return -
latest_risk_free_rate)

print(f"CAPM Expected Return Calculation for Flutter Entertainment")
print(f" Beta (β) vs S&P 500: {beta:.4f}")
```

```
print(f"Risk-Free Rate (US 10Y Treasury): {latest_risk_free_rate:.4%}")
print(f"Market Return (S&P 500): {market_return:.2%}")
print(f"📈 Expected Return (CAPM): {expected_return:.2%} (Annualized)")

Regression Line & Scatterplot of Daily Returns
plt.figure(figsize=(8,5))
plt.scatter(capm_data["Market_Returns"], capm_data["Returns"], alpha=0.5,
label="Daily Returns")
plt.plot(capm_data["Market_Returns"], capm_model.predict(X), color='red',
         label=f"Regression Line (β={beta:.2f})")
plt.xlabel("S&P 500 Daily Returns")
plt.ylabel("Flutter Entertainment Daily Returns")
plt.title("Flutter Beta Calculation vs. S&P 500")
plt.legend()
plt.grid(True)
plt.show()
```

CAPM/Random Forest Prediction

```
import yfinance as yf
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

Historical Data (Using NYSE Listing and USD Data)
stock_ticker = "FLUT"   # Flutter Entertainment (NYSE)
market_ticker = "^GSPC"  # S&P 500 Index (USD)
rf_ticker = "^TNX"       # 10-Year US Treasury Yield (Risk-Free Rate in USD)

# Download Data
stock_data = yf.download(stock_ticker, start="2010-01-01", end="2025-01-01")
market_data = yf.download(market_ticker, start="2010-01-01", end="2025-01-01")
```

```
rf_data = yf.download(rf_ticker, start="2010-01-01", end="2025-01-01")  # Risk-free
rate

# Step 2: Calculate Returns
stock_data['Stock_Returns'] = stock_data['Close'].pct_change()
market_data['Market_Returns'] = market_data['Close'].pct_change()

# Align Data
data = pd.concat([stock_data['Stock_Returns'], market_data['Market_Returns']],
axis=1).dropna()
data.columns = ["Stock_Returns", "Market_Returns"]

beta = 0.67  # Manually defined beta

Expected Return using CAPM
risk_free_rate = float(rf_data['Close'].mean() / 100)  # Convert from percentage
market_return = float(data['Market_Returns'].mean() * 63)  # 3-month expected
market return

expected return using CAPM (3-month horizon)
return_capm = risk_free_rate + beta * (market_return - risk_free_rate)
expected_return_capm = float(expected_return_capm)  # Convert to scalar

CAPM Expected Return
print(f"Expected Return (CAPM, 3 months, β=0.67): {expected_return_capm:.2%}")

Latest Stock Price (Convert to Scalar) Get a cup of tea and continue** 30 mins (
implied price after )
current_price = float(stock_data['Close'].iloc[-1])

# CAPM-Based Implied Price (3-month horizon)
capm_implied_price = float(current_price * (1 + expected_return_capm))
print(f"CAPM-Based Implied Price (3 months, β=0.67): ${capm_implied_price:.2f}")

3
# random forest model
data['Lagged_Market_Returns'] = data['Market_Returns'].shift(1)
data['Lagged_Stock_Returns'] = data['Stock_Returns'].shift(1)
data['Beta'] = beta  # Add Beta as a constant feature
data = data.dropna()

# Define target (stock returns) and features (market returns, lags, and beta)
X = data[['Market_Returns', 'Lagged_Market_Returns', 'Lagged_Stock_Returns',
'Beta']]
y = data['Stock_Returns']

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train a Random Forest Regressor
```

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

Predict Next Period's Return (3-month horizon)
predicted_return_rf = float(rf_model.predict(X_test).mean() * 63)  # Convert to
scalar & 3-month prediction
print(f"Expected Return (Random Forest, 3 months, β=0.67 included):
{predicted_return_rf:.2%}")

# Random Forest-Based Implied Price (3 months)
rf_implied_price = float(current_price * (1 + predicted_return_rf))
print(f"Random Forest-Based Implied Price (3 months, β=0.67 included):
${rf_implied_price:.2f}")

Evaluate the Model
y_pred = rf_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (Random Forest): {mse:.6f}")

Feature Importance
feature_importances = pd.DataFrame({
    'Feature': ['Market_Returns', 'Lagged_Market_Returns', 'Lagged_Stock_Returns',
'Beta'],
    'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances (Random Forest):")
print(feature_importances)
```

Risk Management / Kurtosis

```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

Historical Data
stock_ticker = "FLTR.L"   # Flutter Entertainment (LSE ticker)
ftse_ticker = "^FTSE"     # FTSE 100 Index
sp500_ticker = "^GSPC"    # S&P 500 Index

# Download Data
stock_data = yf.download(stock_ticker, start="2010-01-01", end="2025-01-01")
```

```
ftse_data = yf.download(ftse_ticker, start="2010-01-01", end="2025-01-01")
sp500_data = yf.download(sp500_ticker, start="2010-01-01", end="2025-01-01")

Calculate Returns
stock_data['Stock_Returns'] = stock_data['Close'].pct_change()
ftse_data['FTSE_Returns'] = ftse_data['Close'].pct_change()
sp500_data['SP500_Returns'] = sp500_data['Close'].pct_change()

# Drop NaN values
returns_df = pd.concat([stock_data['Stock_Returns'], ftse_data['FTSE_Returns'],
sp500_data['SP500_Returns']], axis=1).dropna()
returns_df.columns = ["Flutter_Returns", "FTSE_Returns", "SP500_Returns"]

Compute Kurtosis
flutter_kurtosis = stats.kurtosis(returns_df['Flutter_Returns'], fisher=True)
ftse_kurtosis = stats.kurtosis(returns_df['FTSE_Returns'], fisher=True)
sp500_kurtosis = stats.kurtosis(returns_df['SP500_Returns'], fisher=True)

print(f"Kurtosis of Flutter Entertainment: {flutter_kurtosis:.2f}")
print(f"Kurtosis of FTSE 100: {ftse_kurtosis:.2f}")
print(f"Kurtosis of S&P 500: {sp500_kurtosis:.2f}")
Plot Histogram & Compare Distributions
plt.figure(figsize=(8, 5))

# Flutter Entertainment
plt.hist(returns_df['Flutter_Returns'], bins=50, alpha=0.6, color='blue',
edgecolor='black', density=True, label="Flutter Entertainment")

# FTSE 100
plt.hist(returns_df['FTSE_Returns'], bins=50, alpha=0.6, color='green',
edgecolor='black', density=True, label="FTSE 100")

# S&P 500
plt.hist(returns_df['SP500_Returns'], bins=50, alpha=0.6, color='red',
edgecolor='black', density=True, label="S&P 500")

# Add Labels & Title
plt.xlabel("Daily Returns")
plt.ylabel("Density")
plt.title(f"Return Distributions: Flutter vs. FTSE vs. S&P 500\nFlutter Kurtosis:
{flutter_kurtosis:.2f} | FTSE Kurtosis: {ftse_kurtosis:.2f} | S&P 500 Kurtosis:
{sp500_kurtosis:.2f}")
plt.legend()
plt.grid(True)

# Show plot
plt.show()
```

Z-Score

```
import matplotlib.pyplot as plt

# Inputs
current_assets = 24508
total_liabilities = 13241
retained_earnings = 9573
ebit = 1129
market_value_of_equity = 45790
sales = 14708
total_assets = 24508

# Calculate Altman Z-Score components
X1 = (current_assets - total_liabilities) / total_assets
X2 = retained_earnings / total_assets
X3 = ebit / total_assets
X4 = market_value_of_equity / total_liabilities
X5 = sales / total_assets

Z = 1.2 * X1 + 1.4 * X2 + 3.3 * X3 + 0.6 * X4 + 1.0 * X5

risk = "Low Risk (Safe)" if Z > 2.99 else "Moderate Risk (Gray Area)" if Z >= 1.81
else "High Risk (Distress Zone)"


labels = ['X1', 'X2', 'X3', 'X4', 'X5']
values = [1.2 * X1, 1.4 * X2, 3.3 * X3, 0.6 * X4, 1.0 * X5]
goldman_colors = ['#004b87', '#99badd', '#657687', '#e6e9ec', '#b2c7d9']

plt.figure(figsize=(8, 6))
bars = plt.bar(labels, values, color=goldman_colors, alpha=0.8)
plt.title('Altman Z-Score Components', fontsize=14, color='#004b87')
plt.xlabel('Components', fontsize=12, color='#004b87')
plt.ylabel('Weighted Value', fontsize=12, color='#004b87')
plt.axhline(y=Z, color='gray', linestyle='--', label=f"Total Z-Score: {Z:.2f}")
plt.legend(loc='upper left', frameon=False, fontsize=10, title_fontsize=12)

# Add value labels
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, f"{yval:.2f}", ha='center',
va='bottom', color='#004b87')
```

```
plt.tight_layout()
plt.show()
```

Relative Valuation

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# List of tickers (example: Flutter Entertainment and some competitors)
tickers = ['FLTR.L', 'PENN', 'DKNG', 'MGM']

for ticker in tickers:
    try:
        stock = yf.Ticker(ticker)
        info = stock.info
        data[ticker] = {
            'P/E': info.get('forwardPE'),
            'EV/EBITDA': info.get('enterpriseToEbitda'),
            'Price-to-Sales': info.get('priceToSalesTrailing12Months')
        }
    except Exception as e:
        print(f"Could not fetch data for {ticker}: {e}")
```

```
valuation_df = pd.DataFrame(data).T
valuation_df.index.name = 'Company'
valuation_df.dropna(inplace=True)

# Display the table
print(valuation_df)


Blue_colors = ['#004b87', '#99badd', '#657687', '#e6e9ec', '#b2c7d9']

plt.figure(figsize=(10, 6))
for column, color in zip(valuation_df.columns, blue_colors):
    plt.bar(
        valuation_df.index, valuation_df[column], alpha=0.8, label=column,
color=color
    )

plt.title('Relative Valuation Multiples', fontsize=14, color='#004b87')
plt.xlabel('Company', fontsize=12, color='#004b87')
plt.ylabel('Valuation Multiples', fontsize=12, color='#004b87')
plt.xticks(rotation=45, fontsize=10, color='#657687')
plt.legend(title="Metrics", title_fontsize=10, frameon=False)

plt.tight_layout()
plt.show()
```

Options and Implied Volatility

Sourced: Trading View:
https://www.tradingview.com/chart/?symbol=NYSE%3AFLUT

```
from scipy.stats import norm
import math

def black_scholes(S, K, T, r, sigma, option_type="call"):
    """
    Calculate Black-Scholes option price.

    Parameters:
    - S: Current stock price
    - K: Strike price
    - T: Time to expiration (in years)
```

```
    - r: Risk-free interest rate (annualized, decimal form)
    - sigma: Volatility of the stock (annualized, decimal form)
    - option_type: "call" or "put"

    Returns:
    - Black-Scholes price of the option
    """
    d1 = (math.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * math.sqrt(T))
    d2 = d1 - sigma * math.sqrt(T)

    if option_type.lower() == "call":
        price = S * norm.cdf(d1) - K * math.exp(-r * T) * norm.cdf(d2)
    elif option_type.lower() == "put":
        price = K * math.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)
    else:
        raise ValueError("Invalid option_type. Use 'call' or 'put'.")

    return price

def evaluate_option_mispricing(S, K, T, r, market_price, sigma,
option_type="call"):
    """
    Evaluate potential mispricing of the option.

    Parameters:
    - S: Current stock price
    - K: Strike price
    - T: Time to expiration (in years)
    - r: Risk-free interest rate (annualized, decimal form)
    - market_price: Current market price of the option
    - sigma: Implied volatility (converted from % to decimal)
    - option_type: "call" or "put"

    Returns:
    - A dictionary containing theoretical price, market price, and mispricing.
    """
    theoretical_price = black_scholes(S, K, T, r, sigma, option_type)
    mispricing = market_price - theoretical_price

    return {
        "Theoretical Price": round(theoretical_price, 2),
        "Market Price": market_price,
        "Mispricing": round(mispricing, 2),
        "Implied Volatility": sigma
    }

# **Using Real Data from TradingView**
S = 245.53     # Current stock price (from TradingView)
K = 180        # Strike price (from TradingView)
T = 30 / 365   # Time to expiration (approx. 1 month = 0.0822 years)
r = 0.04       # Risk-free rate (assumed 4% annualized)
```

```
market_price = 66.47  # Observed market price of the call option
sigma = 60.91 / 100   # Convert implied volatility from % to decimal

# **Evaluate the Option Mispricing**
result = evaluate_option_mispricing(S, K, T, r, market_price, sigma,
option_type="call")
print(result)


import math

S = 245.53  # Current stock price
IV = 0.6091  # Implied volatility
T = 30 / 365  # Time to expiration in years

expected_move = S * IV * math.sqrt(T)
upper_target = S + expected_move
lower_target = S - expected_move

print(f"Expected Price Range in 1 Month: ${lower_target:.2f} to
${upper_target:.2f}")
```

References

https://www.goldmansachs.com/images/insights/2025-outlooks/Markets-Outlook-2025-Trading-Tails-and-Tailwinds.pdf

https://www.flutter.com

https://www.sec.gov/edgar/browse/?CIK=1635327&owner=exclude (Flutter entertainments 10-K)

https://stablebread.com/altman-z-score/#:~:text=Safe%20Zone%20(Z%2DScore%20%3E,a%20low%20probability%20of%20bankruptcy. (Altman Score, FAJASY, 2024)

https://www.dws.com/en-ie/insights/cio-view/asset-classes/inverted-yield-curves-finally-end-what-now/ (Yield Curves)

Transparent AI Disclosure.

AI-assisted tools were used to format and debug portions of the code to resolve persistent errors and improve readability (fonts, structure). The core logic and implementation remain my own.

- If an error message appeared in my terminal, I used AI tools to figure them out in conjunction with stack overflow.
- After typing my code, I asked AI to format it properly so it can be read an interpreted easier.