Lambton College

STOCK PORTFOLIO RECOMMENDATION AND RISK IDENTIFICATION

USING TIME SERIES FORECASTING & REINFORCEMENT LEARNING

PROJECT MILESTONE- 3 & 4

Project Board, Model Interpretability, Model Tuning, Model System

Presenter : Team Everest

Team Members:
Bharat, Babita, Janvi, Gurpreet, Karmjeet, Valentine, Anmol, Ramandeep, Sarabjot, Lateef

Rationale of the Project

What is the issue?

Investors struggle with stock price volatility, poor risk management, and data overload.

Why is it an issue?

- Inaccurate predictions lead to financial losses.
- Static risk management is ineffective.

Why now?

- Growing demand for personalized financial recommendations
- Economic uncertainty needs real-time insights

How does the project shed light upon this?

- Provides tailored portfolio recommendations.
- Powered by ML algorithms that analyze user preferences and performance metrics.
- Customizable portfolios based on risk levels (low, medium, high).

Milestone 3 & 4 Approach

Project Board

- **Trello Integration:** Public, accessible project board with milestones and tasks.
- **Board Structure:** Organized: To-Do, In-Progress,Done & Trash Bin
- **Benefit:** Tracks progress visually and ensures deadlines are met.

Model Interpretability

- **Tools:** LIME (local explanations)
- **Importance:** Enhances trust, uncovers biases, and improves model transparency.
- **Explanation:** Discuss most influential features and their impact on predictions.

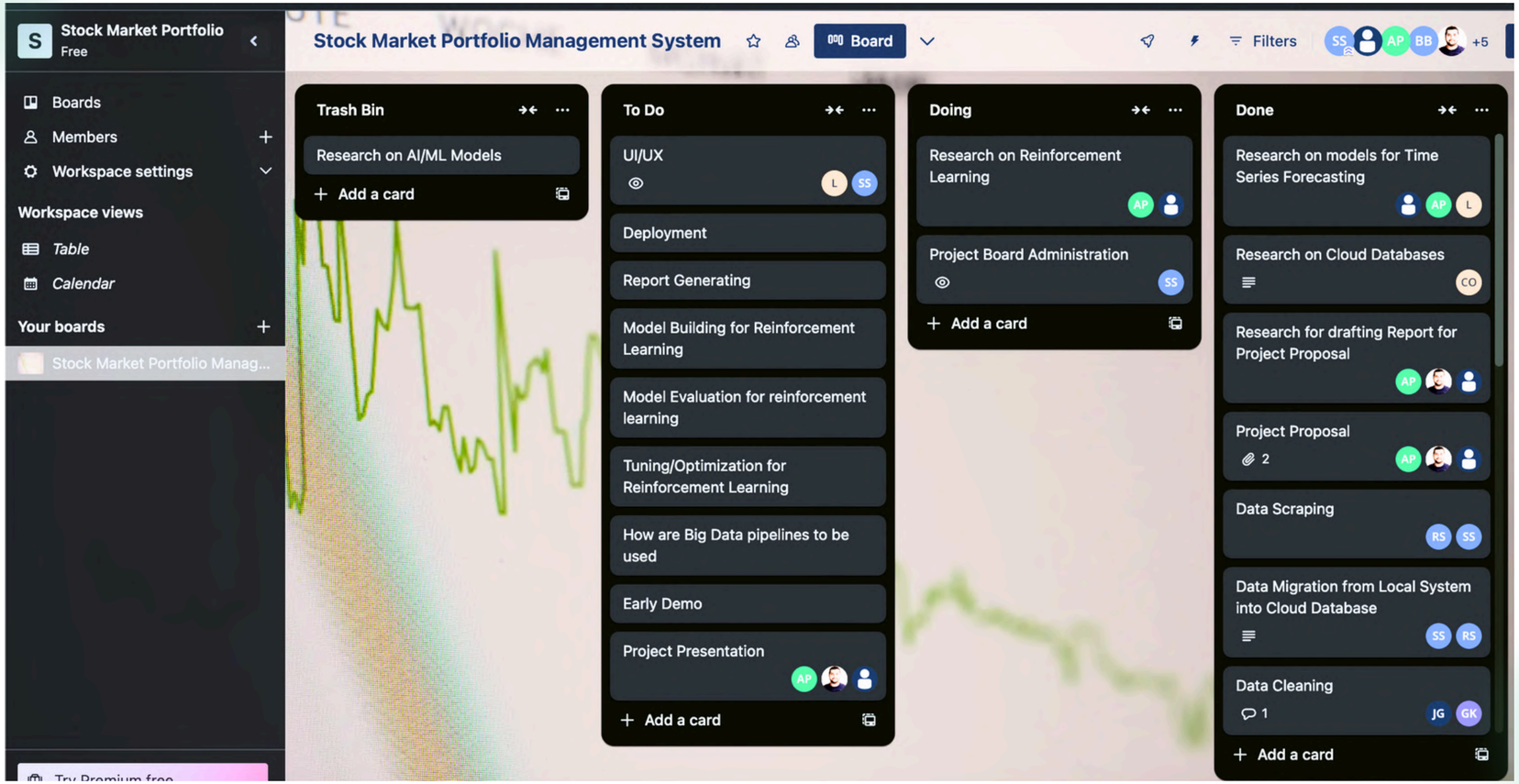
Model Tuning

- **Random Forest Regressor & ARIMA:** Key hyperparameters like *n_estimators* and *max_depth*.
- **Train Validate split & Cross-Validation:** Optimizes parameters for accuracy and generalization.
- **Feature Importance:** Ranks features to enhance interpretability.

Model System

- **Multi-Model Pipeline:** Models working sequentially (e.g., market trend and sector-based predictions).
- **Example Indicators:** RSI, MACD for trend and strength analysis.
- **Output Integration:** Demonstrates combined model outputs for final insights using gradient boosting regressor.

Project Board Overview



- *Purpose:* Track project progress and tasks
- *Tools Used:* Trello (Visual management tool for tasks)
- *Structure:* To-Do (Planned tasks), In Progress (Active tasks), Done (Completed tasks),Trash Bin
- *Benefits:* Visualize workflows, ensure deadlines, organize project steps.

Redirect to Trello Project Board

Feature Engineering (Recap)

```
<class 'pandas.core.frame.DataFrame'>
Index: 455154 entries, 147150 to 609042
Data columns (total 26 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   date                455154 non-null  datetime64[ns, UTC-04:00]
 1   open                455154 non-null  float64
 2   high                455154 non-null  float64
 3   low                 455154 non-null  float64
 4   close               455154 non-null  float64
 5   volume              455154 non-null  int64
 6   dividends           455154 non-null  float64
 7   stock splits        455154 non-null  float64
 8   symbol              455154 non-null  object
 9   capital gains       455154 non-null  float64
 10  previous day close  455154 non-null  float64
 11  rolling_dividends   455154 non-null  float64
 12  rolling_splits      455154 non-null  float64
 13  adj close           455154 non-null  float64
 14  prev_close          455154 non-null  float64
 15  maxdiff             455154 non-null  float64
 16  daydiff             455154 non-null  float64
 17  RSI                 455154 non-null  float64
 18  up                  455154 non-null  float64
 19  dn                  455154 non-null  float64
...
 24  prev_diff           455154 non-null  float64
 25  change_tomorrow     455152 non-null  float64
dtypes: datetime64[ns, UTC-04:00](1), float64(23), int64(1), object(1)
memory usage: 109.9+ MB
```

- *adj close*
- *prev_close*
- *maxdiff*
- *daydiff*
- *change_tomorrow*
- *rolling_dividends*
- *rolling_splits*

Aroon Indicator:

- *Up Indicator: Measures time since the last high in percentage terms.*
- *Down Indicator: Measures time since the last low in percentage terms.*

RSI - Relative Strength Index (Target Variable)

Popular momentum oscillator used in stock analysis to measure the speed and change of price movements. RSI helps to identify overbought and oversold conditions in the market by oscillating between 0 and 100.

$$RSI = 100 - \frac{100}{1 + RS} \qquad RS = \frac{Avg.Gain}{Avg.Loss}$$

Interpretation of RSI Values:

- *RSI above 70: A stock is overbought/overvalued, signaling a potential sell point.*
- *RSI below 30: A stock is oversold, potentially undervalued, and may be a buying opportunity.*

Feature Importance

Feature Importance Analysis - RF Regressor Model

```
# Get feature importances as a list of tuples (column name, importance value)
feature_importances_list = [(numeric_columns[i], float(importance)) for i, importance in enumerate(model.featureImpo

# Create DataFrame from the list of feature importances
importances_df = spark.createDataFrame(feature_importances_list, ["Feature", "Importance"]) \
    .orderBy(col("Importance"), ascending=False)

# Show feature importances
importances_df.show()
```

```
+------------------+--------------------+
|           Feature|          Importance|
+------------------+--------------------+
|                up|  0.4125928767726175|
|                dn|  0.3303327668259434|
|     capital_gains| 0.07338511782933646|
|              MACD| 0.06462853075584815|
|         prev_diff| 0.06160036300550298|
|           daydiff|0.012540378836400246|
| rolling_dividends|0.012278439508470029|
|        SignalLine|0.009430165388200754|
|             close|0.007826710843842549|
|         adj_close|0.006963291976127055|
|             EMA12|0.002572990354764...|
|               low|0.001411980650843712|
|        prev_close|0.001357822267814372|
|              high|0.001164360241819...|
|previous_day_close|7.628574270410179E-4|
|              open|7.032836995436566E-4|
|            volume|1.618630266070473E-4|
|             EMA26|1.383189019135208...|
|           maxdiff|8.960368542244482E-5|
|         dividends|4.810862036690144...|
+------------------+--------------------+
only showing top 20 rows
```

- *Overview: Feature importance values indicate the contribution of each feature to the model's predictive power.*
- *Key Observations*
- *Top Influential Features: maxdiff, prev_day_close, open, dividends*
- *Other Notable Features: volume, EMA26, up, dn*
- *Insight:*
- *This analysis allows us to focus on high-impact features for model refinement.*
- *Helps in reducing model complexity by potentially eliminating low-importance features.*

Model Tuning - Random Forest Regressor

```
# Test set evaluation
predictions = model.transform(test_df)
evaluator = RegressionEvaluator(labelCol="RSI", predictionCol="prediction", metricName="mse")
mse = evaluator.evaluate(predictions)
print(f"Test Mean Squared Error (MSE): {mse}")
```

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.util.SizeEstimator$ (file:/opt/homebrew/Cellar/apache-spark/
3.5.3/libexec/jars/spark-core_2.12-3.5.3.jar) to field java.nio.charset.Charset.name
WARNING: Please consider reporting this to the maintainers of org.apache.spark.util.SizeEstimator$
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
24/11/05 17:28:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
k to expand output; double click to hide output rformance degradation.
24/11/05 17:28:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
24/11/05 17:28:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
24/11/05 17:28:09 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
24/11/05 17:28:09 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
[Stage 36:>                                                          (0 + 1) / 1]

Test Mean Squared Error (MSE): 273.35097142599477
```

Initial MSE for RF

```
# Evaluate the best model from TrainValidationSplit
best_model = tvs_model.bestModel
predictions = best_model.transform(test_df)
mse = evaluator.evaluate(predictions)
print(f"Best Model Mean Squared Error (MSE) after Hyperparameter Tuning: {mse}")
```

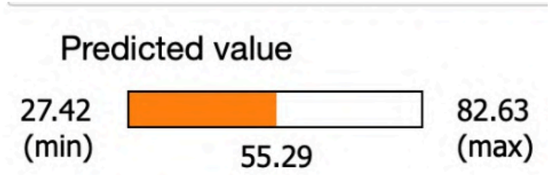
```
24/11/05 21:51:59 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
24/11/05 21:51:59 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
24/11/05 21:51:59 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
24/11/05 21:51:59 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
24/11/05 21:51:59 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partitio
n, this can cause serious performance degradation.
[Stage 68:>                                                          (0 + 1) / 1]

Best Model Mean Squared Error (MSE) after Hyperparameter Tuning: 39.57877651818178
```

MSE after tuning

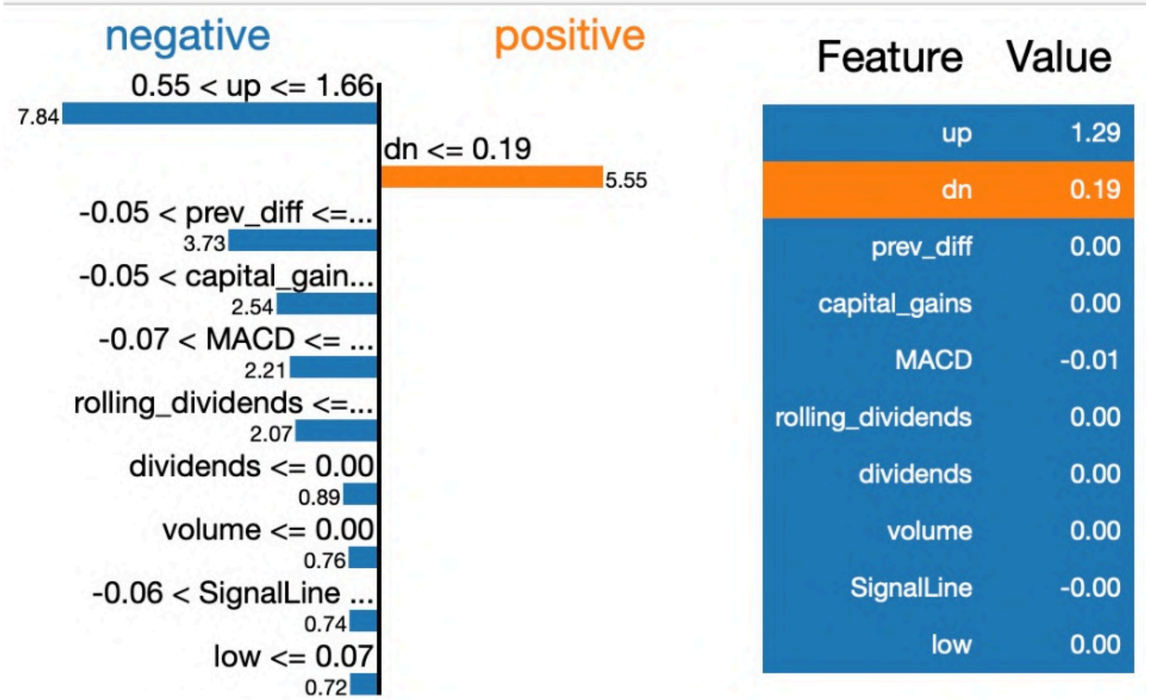
- *Model Performance: Achieved an MSE of 39.099 for the Random Forest Regressor (RFR), indicating strong predictive performance on the test data.*
- *Hyperparameter Tuning: Used TrainValidationSplit (TVS) to optimize parameters, testing combinations of n_estimators, max_depth, and min_samples_split.*
- *Sequential Pattern Capture: Incorporated lagged features and window-based transformations, enabling the model to recognize sequential patterns essential for time-dependent variables like RSI.*

Model Interpretability for RF

Extracting the Influencing Features - using LIME



Predicted Value: 55.29
(Range: 27.42 to 82.63)



Key Feature Contributions:

Positive Impact on Prediction:

dn <= 0.19: Adds 5.55 to the prediction

Key Negative Influencers

0.55 < up <= 1.66: Decreases predicted RSI (Influence: -7.84)
prev_diff <= -0.05: Decreases predicted RSI (Influence: -3.73)
capital_gain <= -0.05: Decreases predicted RSI (Influence: -2.54)
MACD <= -0.07: Decreases predicted RSI (Influence: -2.21)
rolling_dividends <= 0.0: Decreases predicted RSI (Influence: -2.07)

These features are slightly reducing the score, possibly indicating risk factors.

Model Tuning - ARIMA

```
# Split the data into train and test
train_size = int(len(rsi_series) * 0.8)
train, test = rsi_series[:train_size], rsi_series[train_size:]

# Define and fit a basic ARIMA model with default parameters (adjust p, d, q based on initial assumptions)
model_original = ARIMA(train['RSI'], order=(1, 1, 1))
model_original_fit = model_original.fit()

# Forecast using the original model
forecast_original = model_original_fit.forecast(steps=len(test))

# Calculate MSE for the original model
mse_original = mean_squared_error(test['RSI'], forecast_original)
print(f"MSE Before Tuning: {mse_original}")
```

```
C:\Users\kaurp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kaurp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kaurp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kaurp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
MSE Before Tuning: 167.6611587818412
```

Initial MSE for ARIMA

Baseline ARIMA model with default parameters (p=1, d=1, q=1) used as an initial comparison. Result: MSE before tuning is 167.66, with warnings on date frequency and non-invertible parameters.

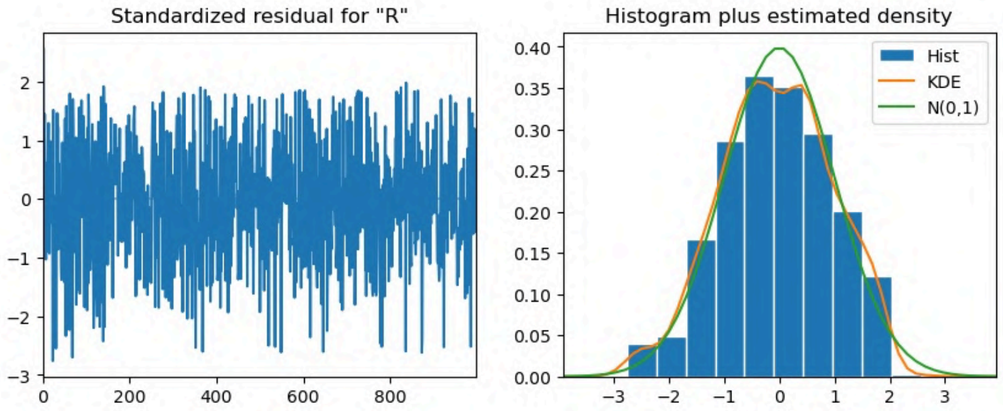
```
# Tuned ARIMA model (after tuning)
# Use the best parameters from tuning (replace with your optimal p, d, q values)
model_tuned = ARIMA(train['RSI'], order=(best_p, best_d, best_q))  # replace best_p, best_d, best_q
model_tuned_fit = model_tuned.fit()

# Forecast using the tuned model
forecast_tuned = model_tuned_fit.forecast(steps=len(test))

# Calculate MSE for the tuned model
mse_tuned = mean_squared_error(test['RSI'], forecast_tuned)
print(f"MSE After Tuning: {mse_tuned}")
```

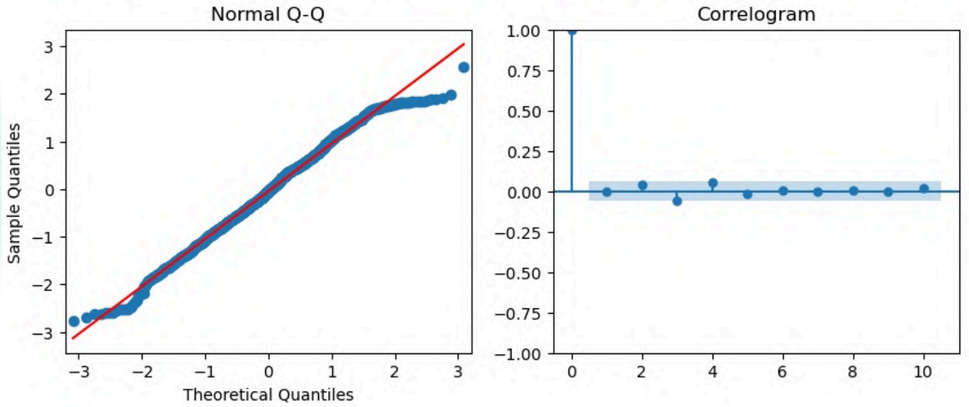
```
C:\Users\kaurp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-
sa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will
forecasting.
  self._init_dates(dates, freq)
C:\Users\kaurp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-
sa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will
forecasting.
  self._init_dates(dates, freq)
C:\Users\kaurp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-
sa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will
forecasting.
  self._init_dates(dates, freq)
MSE After Tuning: 167.84913688848607
```

MSE after tuning

ARIMA model tuned with optimal parameters (best_p, best_d, best_q) and used for forecasting. MSE after tuning is 167.85, with similar warnings about missing date frequency information, which can be ignored if it does not impact the analysis.

Residual Analysis for ARIMA





Standardized Residual Plot (left)

- *Purpose: Detects patterns in residuals over time.*
- *Interpretation: Residuals are randomly scattered around zero.*
- *Conclusion: Indicates no pattern or bias, suggesting a good model fit.*

Histogram with Estimated Density (right)

- *Purpose: Shows residual distribution vs. normal distribution.*
- *Interpretation: Histogram aligns with the normal curve.*
- *Conclusion: Residuals are approximately normally distributed.*

Normal Q-Q Plot (left)

- *Purpose: Checks for normality in residuals.*
- *Interpretation: Points follow the red line closely.*
- *Conclusion: Confirms residuals are normally distributed.*

Correlogram - ACF Plot (right)

- *Purpose: Checks autocorrelation in residuals.*
- *Interpretation: Minimal autocorrelation, except for lag 1.*
- *Conclusion: Residuals are independent, with no systematic errors.*

MODEL SYSTEM

- *Gradient Boosting Regressor (GBR):* *Combined ARIMA and Random Forest predictions to improve accuracy.*
- *Performance Evaluation:* *GBR showed strong performance with RMSE and MSE metrics.*
- *Ensemble Learning:* *Integrated models enhanced prediction accuracy, showcasing the power of ensemble methods.*

```
20]: rmse = evaluator.evaluate(gbt_predictions)
     print(f"Root Mean Squared Error (RMSE) on test data = {rmse}")

     Root Mean Squared Error (RMSE) on test data = 19.47046152344046
```

```
21]: # --- Final Model Output ---
     # You can also evaluate the performance of the GBR model using other metrics, such as MSE.
     gbr_mse = evaluator.evaluate(gbt_predictions, {evaluator.metricName: "mse"})
     print(f"Meta-Model MSE on Test Data: {gbr_mse}")

     Meta-Model MSE on Test Data: 379.0988719357754
```

Conclusion

- Our model achieved a significant accuracy improvement, reducing MSE from 273.35 to 39.099 under RFF.
- For ARIMA, MSE after tuning is 167.85.

Predictions using LIME for both the models (ARIMA and Random Forest Regressor)

This project forms a scalable foundation for broader financial applications, accommodating diverse assets and user preferences.

Challenges of the Project

Model Interpretability

Explaining predictions using SHAP for both the models(ARIMA and Random Forest Regressor) were computationally expensive, so, proceeded with LIME.

Hyperparameter Tuning

Extensive grid search and cross-validation for Random Forest and ARIMA were computationally demanding. So, proceeded with Train Validation Split.

Future Work

Model Deployment Demo

Implement deployment on platforms like Flask or AWS Sagemaker; create an interactive demo via Streamlit or HuggingFace.

Git Repository

Organize code structure, commit history, and branching strategies for transparency.

Final Report

Present an integrated workflow from data processing to final predictions.

Team Everest

Thank You

For Your Attention

Any Queries?

We are open for discussion.

Access this ppt via: https://bit.ly/teameverest2