Applied AI in Business (COM724) Software Prototype and with Technical Report (AE2)

SOLIGENCE CRYPTOCURRENCY AI

student number + Name:	Q102543890 – ANANDHA KRISHNAN MR

TABLE OF CONTENTS

TABLE OF FIGURES

Introduction

Cryptocurrency trading has rapidly emerged as a significant sector in the financial industry, characterized by high volatility, rapid market changes, and immense profit potential. To address these complexities, Solent Intelligence (SOLIGENCE), a global leader in financial technology, sought to develop an Intelligent Cryptocurrency Trading (ICT) platform. This platform leverages advanced machine learning models and interactive data visualization techniques to analyze cryptocurrency trends, predict prices, and provide actionable insights. The ICT platform is designed to empower users with strategic trading decisions by combining sophisticated forecasting algorithms and user-friendly interface elements, ensuring a competitive edge in the dynamic cryptocurrency market.

Statement of the problem

The cryptocurrency market's extreme volatility and lack of accessible tools for informed decisions hinder traditional trading systems' ability to capture intricate data dependencies and patterns, resulting in missed opportunities and poor investment outcomes. Integrated platforms that combine forecasting, correlation analysis, and data visualization are missing, further complicating strategic trading. SOLIGENCE recognized the need for an intelligent system that provides accurate predictions, identifies correlations, and offers actionable insights in an intuitive format.

Aims and Objectives

Design and develop an ICT platform that enhances cryptocurrency trading decisions with intelligent predictions and interactive visualizations. Leverage advanced machine learning models like LSTM, SARIMAX, Random Forest, and Prophet to predict cryptocurrency trends with high accuracy. Integrate real-time data analysis with dynamic visualizations for seamless user engagement. Perform correlation analysis, scenario-based forecasting, and market trend exploration. Provide actionable insights like buy and sell signals to optimize trading strategies. Ensure scalability, adaptability, and usability for diverse datasets, cryptocurrencies, and evolving market conditions.

Need for your prototype

The prototype addresses the critical need for an integrated platform combining predictive modeling, visualization, and actionable insights. Leveraging advanced algorithms and interactive interfaces, it enables users to navigate cryptocurrency trading, optimize investment strategies, and gain a competitive edge. The prototype validates the solution’s feasibility and scalability before full-scale implementation.

Prototype Design

The ICT platform prototype, designed as a modular system, seamlessly integrates machine learning models, visualization tools, and data analysis techniques for cryptocurrency trading. It features a Data Module for fetching and preprocessing historical and real-time data, ensuring clean and reliable inputs. Predictive Models include LSTM for sequential patterns, SARIMAX for seasonal trends, Prophet for long-term forecasts, and Random Forest for feature-based predictions. A Visualization Engine provides interactive plots like exploring temporal trends etc. The GUI Interface, built with Streamlit, offers user-friendly navigation and access to historical data exploration, forecasting, and advanced visualizations, making the platform intuitive and functional.

Proposed Solution

The proposed ICT platform integrates data collection, advanced machine learning models, and interactive visualizations to empower cryptocurrency trading. It supports real-time and historical data analysis, providing accurate forecasts and actionable buy/sell signals. The platform features diverse forecasting models, including LSTM, SARIMAX, Random Forest, and Prophet, ensuring robust predictive capabilities. An interactive Streamlit GUI offers a user-friendly interface for exploring data, predictions, and correlation analyses. The system is scalable, adapting to diverse datasets, cryptocurrencies, and market conditions, making it suitable for long-term applications.

Prototype Development and AI Algorithms used

Utilizing advanced AI and machine learning tools, I built the prototype for this project. The development was done in the Anaconda Environment with Jupyter Notebook for coding and testing. Python 3.10 ensured compatibility with the latest libraries and frameworks for data analysis, modeling, and visualization. This combination provided a robust environment for developing the Intelligent Cryptocurrency Trading (ICT) platform.

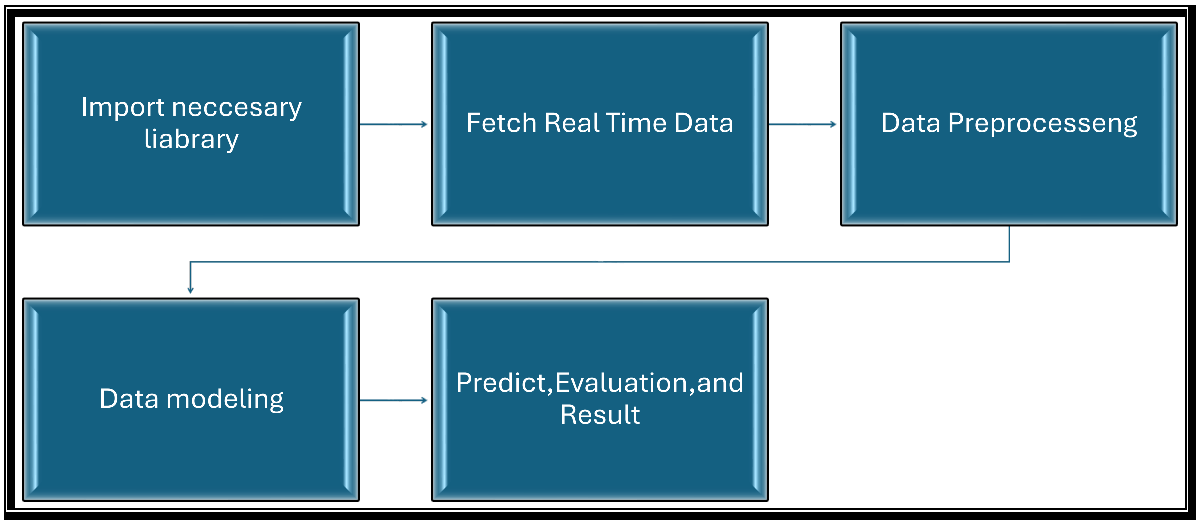
Table Below showcases the Steps Taken to Develop the Software Modules:

Step Description	**Module**
Collected and preprocessed real-time and historical cryptocurrency data from various sources (e.g., Yahoo Finance).	data_fetcher.py
Performed feature engineering, including moving averages, lagged features, and volatility metrics.	data_preprocessing.py
Built and trained the LSTM model to predict cryptocurrency prices using sequential data.	lstm_forecasting_module.py
Developed a SARIMAX model to account for seasonality and exogenous variables for robust forecasting.	sarimax_forecasting.py
Used the Random Forest model for feature-based regression to predict future prices.	random_forest_crypto.py
Implemented the Prophet model for long-term trend forecasting with additional regressors (e.g., volume, moving averages).	prophet_forecasting_module.py
Conducted exploratory data analysis (EDA), including temporal trends, KDE plots, and volatility analysis.	crypto_visualizations.py
Performed clustering and correlation analysis to group cryptocurrencies and identify interdependencies.	correlation_analysis.py
Designed a Streamlit-based GUI for seamless interaction with data visualizations, predictions, and analyses.	lstm_forecasting_module.py, random_forest_crypto.py, sarimax_forecasting.py, prophet_forecasting_module.py
Created a candlestick chart and correlation heatmap to enhance visualization of market trends and relationships.	crypto_visualizations.py

The Table below showcases the libraries and packages used:

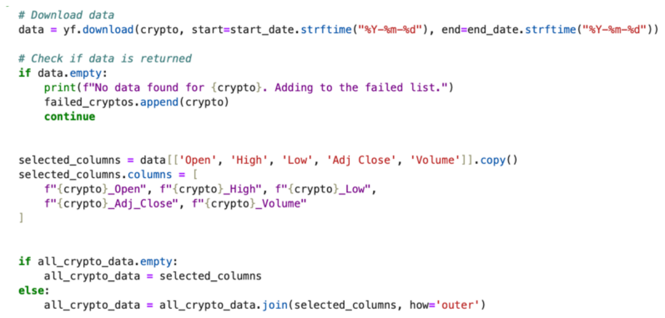
Libraries	Description
NumPy	Provides support for handling arrays, numerical computations, and feature engineering tasks.
Pandas	Used for data manipulation, cleaning, and preprocessing of cryptocurrency datasets.
Matplotlib	Generates static plots for data visualizations, including correlation matrices and trend charts.
Plotly	Creates interactive visualizations, such as candlestick charts, moving averages, and forecasts.
Seaborn	Used for creating heatmaps and advanced statistical plots for correlation analysis.
Scikit-learn	Provides tools for preprocessing, feature scaling, and implementing Random Forest models.
Statsmodels	Utilized for SARIMAX modeling to handle seasonality and exogenous variables in time series.
Prophet	Used for long-term forecasting with added support for custom seasonalities and regressors.
TensorFlow/Keras	Builds and trains the LSTM model for sequential cryptocurrency price prediction.
Streamlit	Powers the GUI for user interaction with cryptocurrency analysis, forecasting, and visualizations.
Datetime	Handles date and time operations, including generating time ranges for data analysis.
Random	Generates random values for tasks like sampling cluster representatives or responses.

Workflow of the project



Data Collection and Pre-Processing

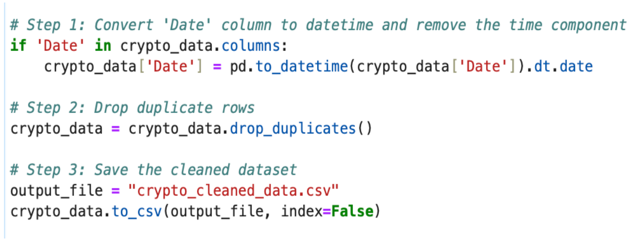
The Intelligent Coin Trading (ICT) platform's development started with collecting and preprocessing cryptocurrency data. Using Yahoo Finance APIs through the Python library yfinance, the dataset was created with historical and real-time data for 30 cryptocurrencies, including Bitcoin, Ethereum, and Litecoin. The script downloaded variables like Open, High, Low, Adjusted Close, and Volume, and handled missing data.



```python
# Download data
data = yf.download(crypto, start=start_date.strftime("%Y-%m-%d"), end=end_date.strftime("%Y-%m-%d"))

# Check if data is returned
if data.empty:
    print(f"No data found for {crypto}. Adding to the failed list.")
    failed_cryptos.append(crypto)
    continue


selected_columns = data[['Open', 'High', 'Low', 'Adj Close', 'Volume']].copy()
selected_columns.columns = [
    f"{crypto}_Open", f"{crypto}_High", f"{crypto}_Low",
    f"{crypto}_Adj_Close", f"{crypto}_Volume"
]


if all_crypto_data.empty:
    all_crypto_data = selected_columns
else:
    all_crypto_data = all_crypto_data.join(selected_columns, how='outer')
```

Figure 1 Downloading the Data

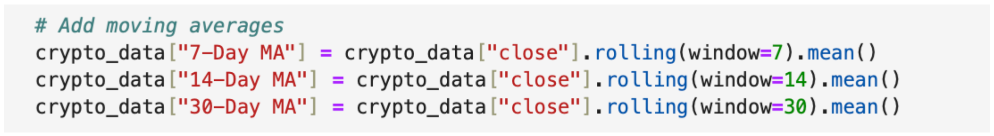
The collected data was saved to a CSV file (crypto_complete_data_1_year.csv) for reuse. It was then processed to remove duplicates, standardize date formatting, and save as crypto_cleaned_data.csv for model training and prediction.



```python
# Step 1: Convert 'Date' column to datetime and remove the time component
if 'Date' in crypto_data.columns:
    crypto_data['Date'] = pd.to_datetime(crypto_data['Date']).dt.date

# Step 2: Drop duplicate rows
crypto_data = crypto_data.drop_duplicates()

# Step 3: Save the cleaned dataset
output_file = "crypto_cleaned_data.csv"
crypto_data.to_csv(output_file, index=False)
```

Figure 2 Crypto Cleaned File

To support real-time analysis, a dynamic function (get_crypto_data_realtime) fetches, cleans, and preprocesses live cryptocurrency data on demand. It validates symbols, downloads data within a specified date range, and ensures data integrity. Key steps include renaming columns, ensuring numerical formats, and computing moving averages (7-day, 14-day, and 30-day) for trend analysis.



```python
# Add moving averages
crypto_data["7-Day MA"] = crypto_data["close"].rolling(window=7).mean()
crypto_data["14-Day MA"] = crypto_data["close"].rolling(window=14).mean()
crypto_data["30-Day MA"] = crypto_data["close"].rolling(window=30).mean()
```

Figure 3 Adding Moving Averages

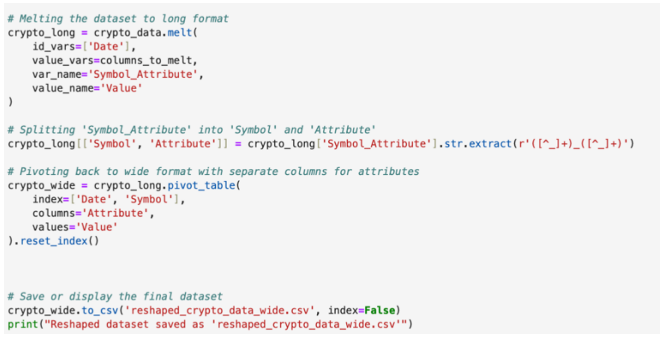
The resulting real-time dataset provided clean, structured data with critical features for model training, including closing prices, volume, and computed moving averages. Smoothing out short-term fluctuations with moving averages enhanced the data's predictive capabilities. This phase ensured high-quality, preprocessed data for developing predictive models and supporting the ICT platform's decision-making.

Data Transformation and Clustering for Software Development

Reshaping, transforming, and clustering the collected cryptocurrency data was crucial for structuring it to support advanced analytics like ARIMA and LSTM models for prediction and clustering algorithms to group cryptocurrencies based on characteristics.

Reshaping and Cleaning Data

The dataset was reshaped to structure data for prediction and analysis. Using the melt function, it was transformed from wide to long format, splitting column names into Symbol (cryptocurrency names) and Attribute. This made data manipulation and visualization easier for each cryptocurrency. The reshaped data was saved as reshaped_crypto_data_wide.csv for building predictive models and visualization.

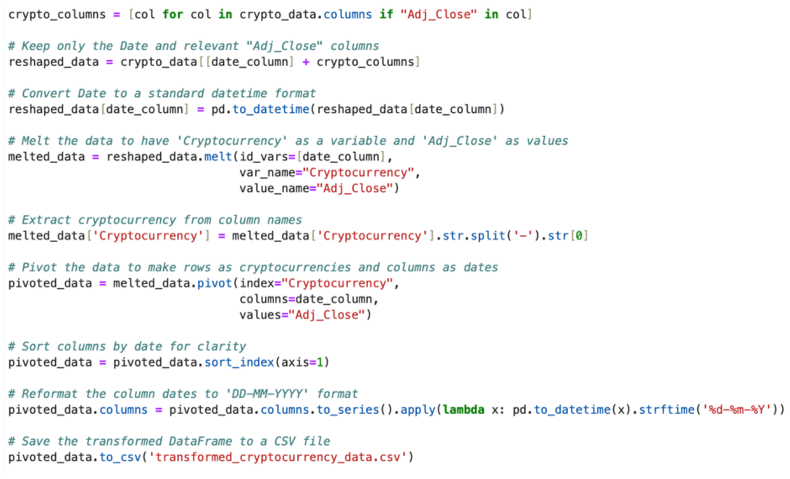


```python
# Melting the dataset to long format
crypto_long = crypto_data.melt(
    id_vars=['Date'],
    value_vars=columns_to_melt,
    var_name='Symbol_Attribute',
    value_name='Value'
)

# Splitting 'Symbol_Attribute' into 'Symbol' and 'Attribute'
crypto_long[['Symbol', 'Attribute']] = crypto_long['Symbol_Attribute'].str.extract(r'([^_]+)_([^_]+)')

# Pivoting back to wide format with separate columns for attributes
crypto_wide = crypto_long.pivot_table(
    index=['Date', 'Symbol'],
    columns='Attribute',
    values='Value'
).reset_index()



# Save or display the final dataset
crypto_wide.to_csv('reshaped_crypto_data_wide.csv', index=False)
print("Reshaped dataset saved as 'reshaped_crypto_data_wide.csv'")
```

Figure 4 Reshaped Dataset

Transforming Data for Clustering

The dataset was transformed to pivot by cryptocurrency names and dates, isolating Adj_Close values. Dates were reformatted, and missing values interpolated linearly for smooth data. The resulting transformed_cryptocurrency_data.csv dataset was used for clustering and grouping cryptocurrencies.

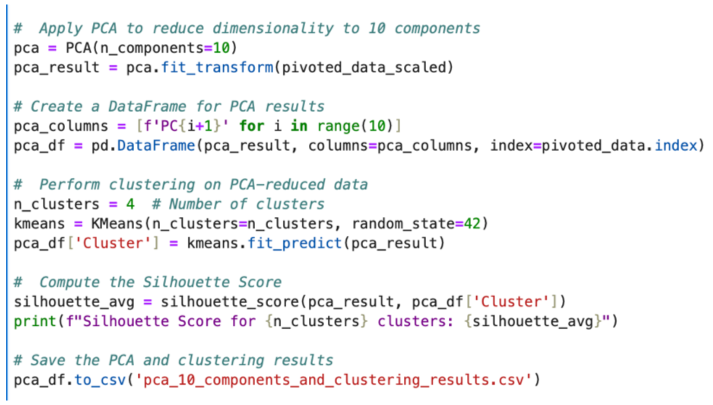


```python
crypto_columns = [col for col in crypto_data.columns if "Adj_Close" in col]

# Keep only the Date and relevant "Adj_Close" columns
reshaped_data = crypto_data[[date_column] + crypto_columns]

# Convert Date to a standard datetime format
reshaped_data[date_column] = pd.to_datetime(reshaped_data[date_column])

# Melt the data to have 'Cryptocurrency' as a variable and 'Adj_Close' as values
melted_data = reshaped_data.melt(id_vars=[date_column],
                                 var_name="Cryptocurrency",
                                 value_name="Adj_Close")

# Extract cryptocurrency from column names
melted_data['Cryptocurrency'] = melted_data['Cryptocurrency'].str.split('-').str[0]

# Pivot the data to make rows as cryptocurrencies and columns as dates
pivoted_data = melted_data.pivot(index="Cryptocurrency",
                                 columns=date_column,
                                 values="Adj_Close")

# Sort columns by date for clarity
pivoted_data = pivoted_data.sort_index(axis=1)

# Reformat the column dates to 'DD-MM-YYYY' format
pivoted_data.columns = pivoted_data.columns.to_series().apply(lambda x: pd.to_datetime(x).strftime('%d-%m-%Y'))

# Save the transformed DataFrame to a CSV file
pivoted_data.to_csv('transformed_cryptocurrency_data.csv')
```

Figure 5 Transformed Dataset For clustering

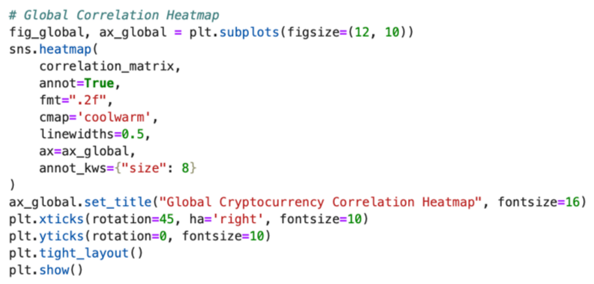
Dimensionality Reduction, Clustering, and Correlation Analysis

To identify patterns and group similar cryptocurrencies, the transformed dataset underwent dimensionality reduction using PCA. Standardized data was reduced to 10 principal components and clustered into 4 groups using K-Means. The silhouette score evaluated clustering quality. Visualizations using the first two PCA components interpreted cryptocurrency groupings. A global correlation matrix identified strong positive and negative correlations, visualized as a heatmap. Specific correlation analyses for selected cryptocurrencies revealed their top positive and negative correlated counterparts, supporting strategic decision-making and potential trading pairs and diversification opportunities.



```
#  Apply PCA to reduce dimensionality to 10 components
pca = PCA(n_components=10)
pca_result = pca.fit_transform(pivoted_data_scaled)

# Create a DataFrame for PCA results
pca_columns = [f'PC{i+1}' for i in range(10)]
pca_df = pd.DataFrame(pca_result, columns=pca_columns, index=pivoted_data.index)

#  Perform clustering on PCA-reduced data
n_clusters = 4  # Number of clusters
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
pca_df['Cluster'] = kmeans.fit_predict(pca_result)

#  Compute the Silhouette Score
silhouette_avg = silhouette_score(pca_result, pca_df['Cluster'])
print(f"Silhouette Score for {n_clusters} clusters: {silhouette_avg}")

# Save the PCA and clustering results
pca_df.to_csv('pca_10_components_and_clustering_results.csv')
```

Figure 6 PCA and K-Means



```
# Global Correlation Heatmap
fig_global, ax_global = plt.subplots(figsize=(12, 10))
sns.heatmap(
    correlation_matrix,
    annot=True,
    fmt=".2f",
    cmap='coolwarm',
    linewidths=0.5,
    ax=ax_global,
    annot_kws={"size": 8}
)
ax_global.set_title("Global Cryptocurrency Correlation Heatmap", fontsize=16)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(rotation=0, fontsize=10)
plt.tight_layout()
plt.show()
```

Figure 7 Global Correlation Heatmap

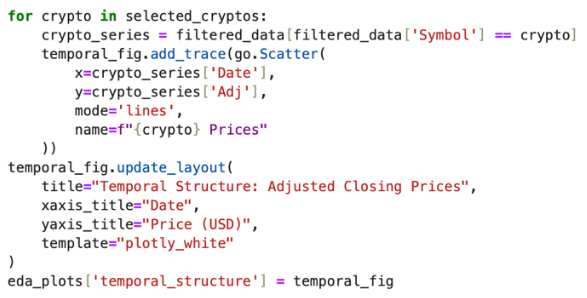
The reshaped dataset played a key role in predictive modeling, especially for time-series forecasting using ARIMA and LSTM models. It accurately predicted cryptocurrency price trends and supported advanced time-series analysis. The transformed dataset also enabled clustering cryptocurrencies based on Adjusted Close trends, aiding strategic decision-making and potential diversification. These processes prepared the data for advanced predictive and clustering models, enhancing the ICT platform's capability to deliver actionable insights for intelligent cryptocurrency trading.

Exploratory Data Analysis (EDA) and Visualization

The next step in ICT platform development was building an interactive software module for Exploratory Data Analysis (EDA) on selected cryptocurrencies. Users gained dynamic insights into market behavior through advanced visualizations and statistical analyses. Using libraries like pandas, plotly, and statsmodels, the EDA module explored temporal structures, price distributions, volatility patterns, and more.

Temporal Structure Analysis

Interactive line plots displayed the adjusted closing prices of selected cryptocurrencies over time, allowing users to observe price trends and temporal variations.

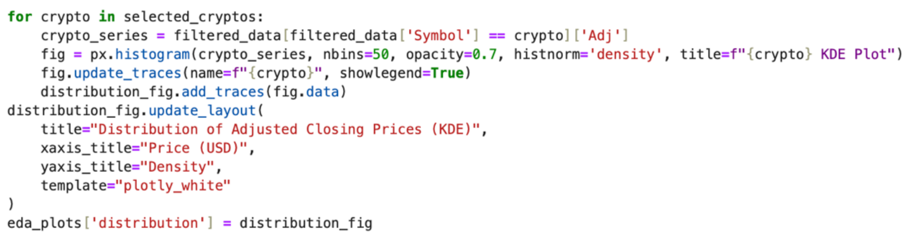


```
for crypto in selected_cryptos:
    crypto_series = filtered_data[filtered_data['Symbol'] == crypto]
    temporal_fig.add_trace(go.Scatter(
        x=crypto_series['Date'],
        y=crypto_series['Adj'],
        mode='lines',
        name=f"{crypto} Prices"
    ))
temporal_fig.update_layout(
    title="Temporal Structure: Adjusted Closing Prices",
    xaxis_title="Date",
    yaxis_title="Price (USD)",
    template="plotly_white"
)
eda_plots['temporal_structure'] = temporal_fig
```

Figure 8 Temporal Variations Plot

Price Distribution

Kernel Density Estimation (KDE) plots provided insights into the probability distribution of adjusted closing prices, helping users understand the overall price behavior of cryptocurrencies.

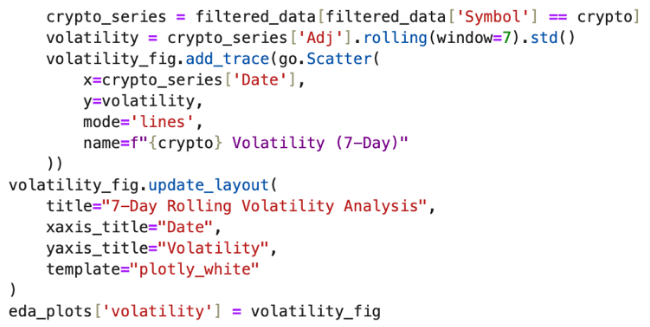


```
for crypto in selected_cryptos:
    crypto_series = filtered_data[filtered_data['Symbol'] == crypto]['Adj']
    fig = px.histogram(crypto_series, nbins=50, opacity=0.7, histnorm='density', title=f"{crypto} KDE Plot")
    fig.update_traces(name=f"{crypto}", showlegend=True)
    distribution_fig.add_traces(fig.data)
distribution_fig.update_layout(
    title="Distribution of Adjusted Closing Prices (KDE)",
    xaxis_title="Price (USD)",
    yaxis_title="Density",
    template="plotly_white"
)
eda_plots['distribution'] = distribution_fig
```

Figure 9 Price Distribution Plot

Rolling Volatility Analysis

Rolling standard deviation plots visualized the short-term price volatility of cryptocurrencies, offering users a view of risk patterns over a 7-day window.

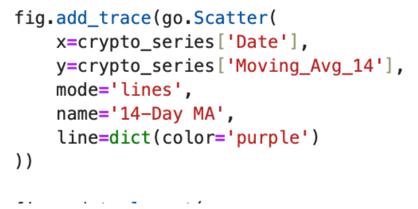


```
        crypto_series = filtered_data[filtered_data['Symbol'] == crypto]
        volatility = crypto_series['Adj'].rolling(window=7).std()
        volatility_fig.add_trace(go.Scatter(
            x=crypto_series['Date'],
            y=volatility,
            mode='lines',
            name=f"{crypto} Volatility (7-Day)"
        ))
    volatility_fig.update_layout(
        title="7-Day Rolling Volatility Analysis",
        xaxis_title="Date",
        yaxis_title="Volatility",
        template="plotly_white"
    )
    eda_plots['volatility'] = volatility_fig
```

Figure 10 Rolling Volatility Plot

Candlestick Charts with Moving Averages

Candlestick charts were integrated to visualize the Open, High, Low, and Adjusted Close (OHLC) prices. Overlayed moving averages (7-day and 14-day) added trend analysis capabilities.

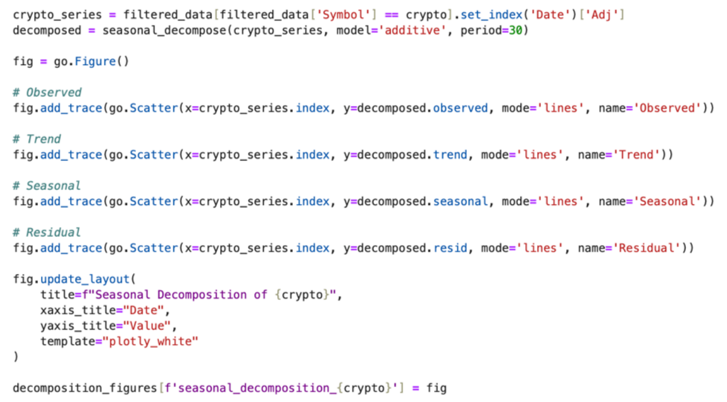


```
fig.add_trace(go.Scatter(
    x=crypto_series['Date'],
    y=crypto_series['Moving_Avg_14'],
    mode='lines',
    name='14-Day MA',
    line=dict(color='purple')
))
```

Figure 11 Moving Average Plot

Seasonal Decomposition

Seasonal decomposition plots broke down the time-series data into observed, trend, seasonal, and residual components, helping users identify recurring patterns and anomalies.

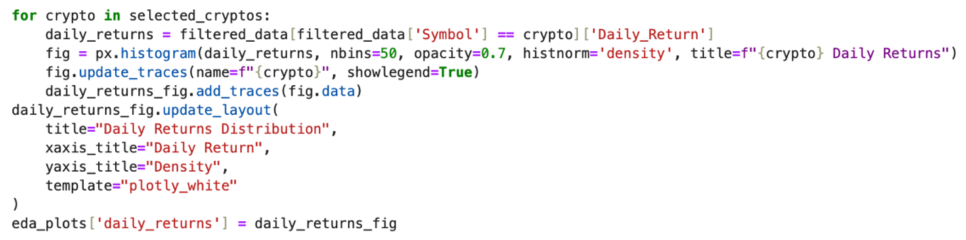


```python
crypto_series = filtered_data[filtered_data['Symbol'] == crypto].set_index('Date')['Adj']
decomposed = seasonal_decompose(crypto_series, model='additive', period=30)

fig = go.Figure()

# Observed
fig.add_trace(go.Scatter(x=crypto_series.index, y=decomposed.observed, mode='lines', name='Observed'))

# Trend
fig.add_trace(go.Scatter(x=crypto_series.index, y=decomposed.trend, mode='lines', name='Trend'))

# Seasonal
fig.add_trace(go.Scatter(x=crypto_series.index, y=decomposed.seasonal, mode='lines', name='Seasonal'))

# Residual
fig.add_trace(go.Scatter(x=crypto_series.index, y=decomposed.resid, mode='lines', name='Residual'))

fig.update_layout(
    title=f"Seasonal Decomposition of {crypto}",
    xaxis_title="Date",
    yaxis_title="Value",
    template="plotly_white"
)

decomposition_figures[f'seasonal_decomposition_{crypto}'] = fig
```

Figure 12 Seasonal Plots

Daily Returns Distribution

Histograms of daily returns allowed users to evaluate the volatility and risk associated with their selected cryptocurrencies.



```python
for crypto in selected_cryptos:
    daily_returns = filtered_data[filtered_data['Symbol'] == crypto]['Daily_Return']
    fig = px.histogram(daily_returns, nbins=50, opacity=0.7, histnorm='density', title=f"{crypto} Daily Returns")
    fig.update_traces(name=f"{crypto}", showlegend=True)
    daily_returns_fig.add_traces(fig.data)
daily_returns_fig.update_layout(
    title="Daily Returns Distribution",
    xaxis_title="Daily Return",
    yaxis_title="Density",
    template="plotly_white"
)
eda_plots['daily_returns'] = daily_returns_fig
```

Figure 13 Daily Returns Plot

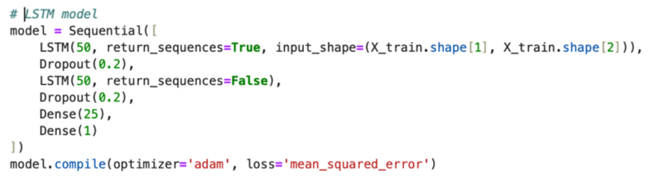
This EDA module provided comprehensive analysis and visualization of cryptocurrency data, enabling users to understand market dynamics. Interactive plots and statistical insights supported the platform’s prediction and decision-support features, empowering informed trading decisions.

Machine Learning Models for Cryptocurrency Prediction

Multiple machine learning models were implemented in the final stage of developing the ICT platform to forecast cryptocurrency prices and provide actionable insights like buy and sell signals. Each model handled different aspects of the cryptocurrency market, combining advanced forecasting techniques with interactive visualizations. The following models were developed and integrated into the platform.

Long Short-Term Memory (LSTM) Neural Network

The LSTM model captured temporal dependencies and forecasted future cryptocurrency prices based on historical data. Preprocessing included feature engineering for moving averages and Min Max Scaling. The model had two LSTM layers, dropout layers, and dense layers. It predicted price ranges and buy/sell signals based on moving averages.

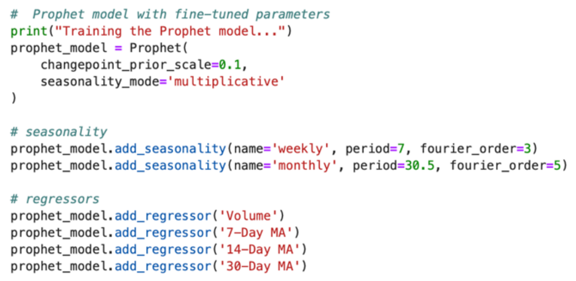


```
# LSTM model
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
    Dropout(0.2),
    LSTM(50, return_sequences=False),
    Dropout(0.2),
    Dense(25),
    Dense(1)
])
model.compile(optimizer='adam', loss='mean_squared_error')
```

Figure 14 LSTM Model

Prophet Model

Prophet, a time-series forecasting library, was fine-tuned for cryptocurrency prediction. It incorporated moving averages and volume as additional regressors to improve accuracy. The model predicted future prices and generated buy/sell signals using moving average crossovers. Prophet’s seasonal components captured cryptocurrency data’s inherent periodicity.

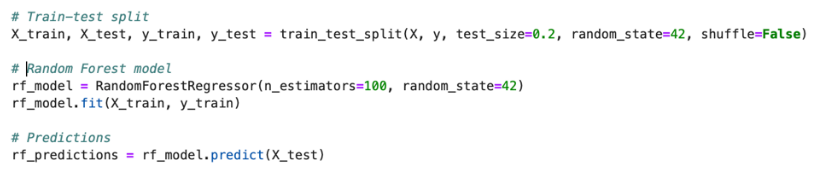


```
#  Prophet model with fine-tuned parameters
print("Training the Prophet model...")
prophet_model = Prophet(
    changepoint_prior_scale=0.1,
    seasonality_mode='multiplicative'
)

# seasonality
prophet_model.add_seasonality(name='weekly', period=7, fourier_order=3)
prophet_model.add_seasonality(name='monthly', period=30.5, fourier_order=5)

# regressors
prophet_model.add_regressor('Volume')
prophet_model.add_regressor('7-Day MA')
prophet_model.add_regressor('14-Day MA')
prophet_model.add_regressor('30-Day MA')
```

Figure 15 Prophet Model

Random Forest

The Random Forest model, an ensemble learning technique, used engineered features like moving averages, lagged values, and volatility to predict cryptocurrency prices. It also identified market trends and generated buy/sell signals based on its predictions.



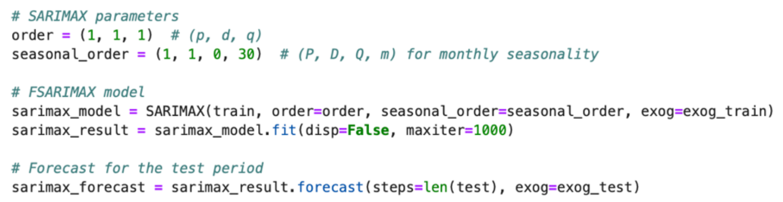
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, shuffle=False)

# Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
rf_predictions = rf_model.predict(X_test)
```

Figure 16 Random Forest Model

SARIMAX (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables)

The SARIMAX model, robust to time-series forecasting, incorporated seasonality and exogenous variables. It used moving averages, lagged values, and volatility to capture complex price trends. The model generated buy and sell signals, plotted with predicted prices and moving averages for interpretability.

These models enhanced the ICT platform’s predictive capabilities, offering users a comprehensive tool for forecasting cryptocurrency prices. Interactive visualizations and actionable insights empowered users to make informed trading decisions, solidifying its role as a decision-support system for cryptocurrency investment.

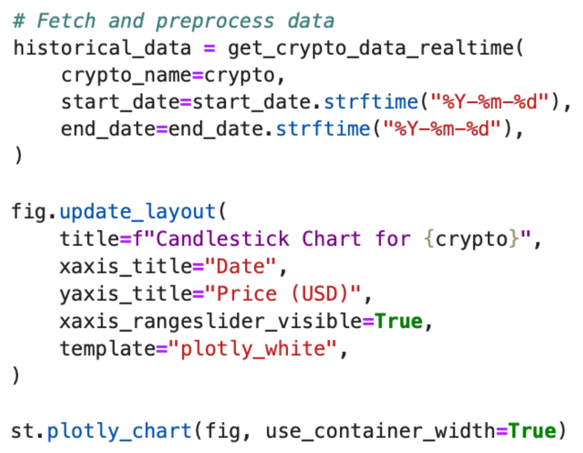


```
# SARIMAX parameters
order = (1, 1, 1)  # (p, d, q)
seasonal_order = (1, 1, 0, 30)  # (P, D, Q, m) for monthly seasonality

# FSARIMAX model
sarimax_model = SARIMAX(train, order=order, seasonal_order=seasonal_order, exog=exog_train)
sarimax_result = sarimax_model.fit(disp=False, maxiter=1000)

# Forecast for the test period
sarimax_forecast = sarimax_result.forecast(steps=len(test), exog=exog_test)
```

Figure 17 SARIMAX Model

Integration of Models and Visualization into a GUI

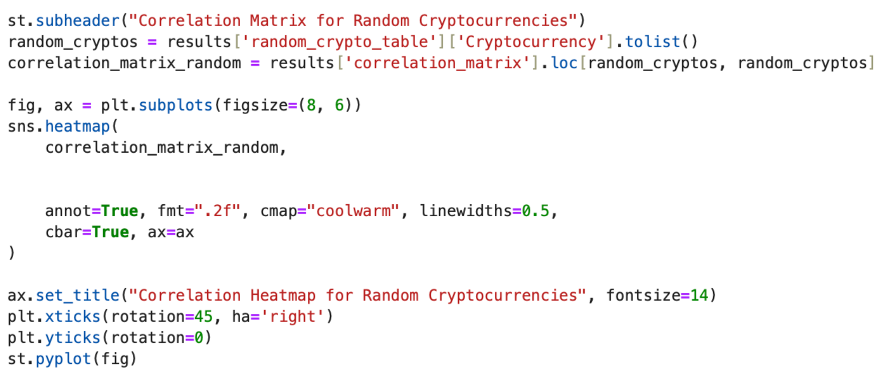
The final phase of ICT platform development integrated predictive models, data visualizations, and analytical tools into an interactive Streamlit GUI. Users can explore historical data, perform correlation analysis, and access advanced forecasting models. The home page has a visually engaging interface, while functionalities are accessible via a sidebar menu. The Historical Data Viewer lets users fetch cryptocurrency data for specified date ranges, visualize trends with interactive charts, and display moving averages.



```python
# Fetch and preprocess data
historical_data = get_crypto_data_realtime(
    crypto_name=crypto,
    start_date=start_date.strftime("%Y-%m-%d"),
    end_date=end_date.strftime("%Y-%m-%d"),
)

fig.update_layout(
    title=f"Candlestick Chart for {crypto}",
    xaxis_title="Date",
    yaxis_title="Price (USD)",
    xaxis_rangeslider_visible=True,
    template="plotly_white",
)

st.plotly_chart(fig, use_container_width=True)
```

Figure 18 Historical Data Plot

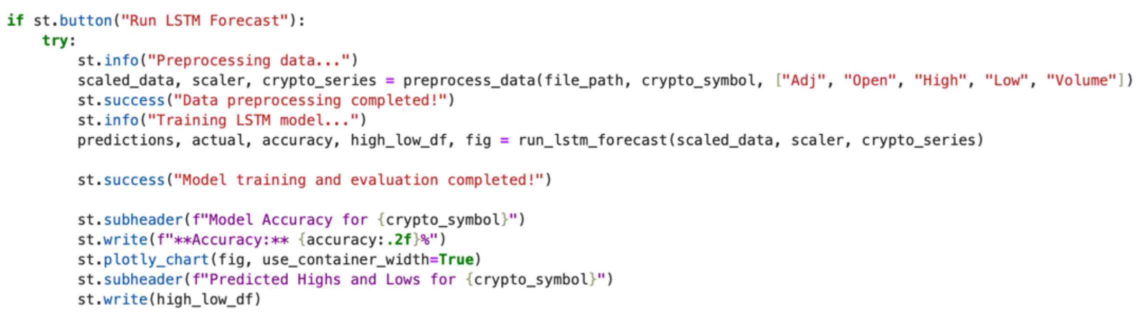
The Correlation Analysis feature enables clustering and visualization of interdependencies among cryptocurrencies using heatmaps and clustering models.



```
st.subheader("Correlation Matrix for Random Cryptocurrencies")
random_cryptos = results['random_crypto_table']['Cryptocurrency'].tolist()
correlation_matrix_random = results['correlation_matrix'].loc[random_cryptos, random_cryptos]

fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(
    correlation_matrix_random,


    annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5,
    cbar=True, ax=ax
)

ax.set_title("Correlation Heatmap for Random Cryptocurrencies", fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
st.pyplot(fig)
```

Figure 19 Correlation Plot

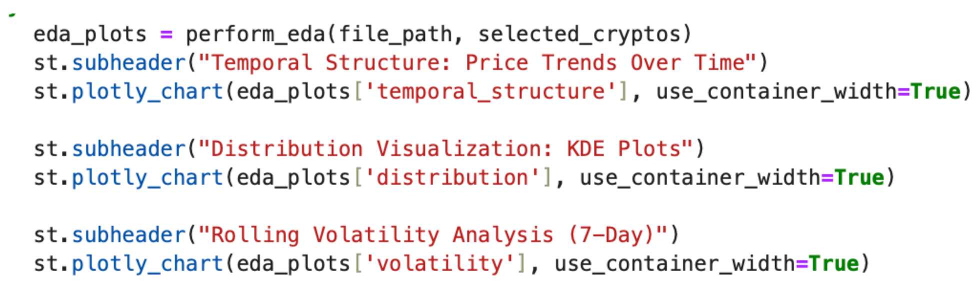
The GUI integrates forecasting models like LSTM, SARIMAX, Random Forest, and Prophet, enabling users to analyze predictions, accuracy, and buy/sell signals. For instance, LSTM generates predictions visualized with actual data and moving averages.



```
if st.button("Run LSTM Forecast"):
    try:
        st.info("Preprocessing data...")
        scaled_data, scaler, crypto_series = preprocess_data(file_path, crypto_symbol, ["Adj", "Open", "High", "Low", "Volume"])
        st.success("Data preprocessing completed!")
        st.info("Training LSTM model...")
        predictions, actual, accuracy, high_low_df, fig = run_lstm_forecast(scaled_data, scaler, crypto_series)

        st.success("Model training and evaluation completed!")

        st.subheader(f"Model Accuracy for {crypto_symbol}")
        st.write(f"**Accuracy:** {accuracy:.2f}%")
        st.plotly_chart(fig, use_container_width=True)
        st.subheader(f"Predicted Highs and Lows for {crypto_symbol}")
        st.write(high_low_df)
```

Figure 20 LSTM Model plot

Users can create dynamic visualizations for temporal trends, volatility, and seasonal decomposition to better understand cryptocurrency market behaviors. The platform's comprehensive integration makes it highly accessible and interactive, empowering users to analyze, predict, and strategize effectively.



```
eda_plots = perform_eda(file_path, selected_cryptos)
st.subheader("Temporal Structure: Price Trends Over Time")
st.plotly_chart(eda_plots['temporal_structure'], use_container_width=True)

st.subheader("Distribution Visualization: KDE Plots")
st.plotly_chart(eda_plots['distribution'], use_container_width=True)

st.subheader("Rolling Volatility Analysis (7-Day)")
st.plotly_chart(eda_plots['volatility'], use_container_width=True)
```

Figure 21 EDA Plots

How to run Code:

To run the software, follow these steps:

1. Prepare your environment.
2. Install necessary libraries.
3. Download the project files.
4. Extract the ZIP file and open the project folder in Jupyter Notebook.
5. Find the app.py file and open it in your terminal. (conda activate environment) Mac OS eg : cd /Users/anandhu336/Downloads/AE2_SOFTWARE_COM724
6. Run the app: streamlit run app.py

Evaluation

The prototype underwent rigorous evaluation to ensure effectiveness and reliability. Machine learning models achieved high accuracy, as evidenced by MAPE. The GUI interface received positive feedback for its intuitive design. Performance benchmarking confirmed alignment with industry standards. Scalability was demonstrated by adapting seamlessly to different cryptocurrencies and datasets, ensuring long-term versatility.

Output

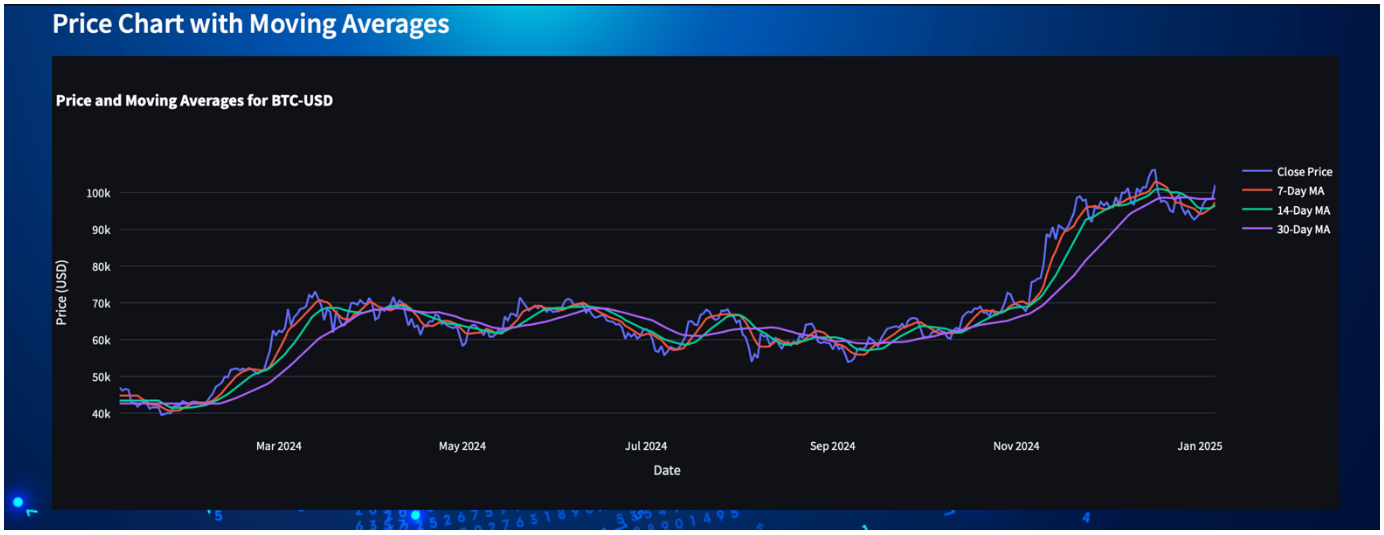
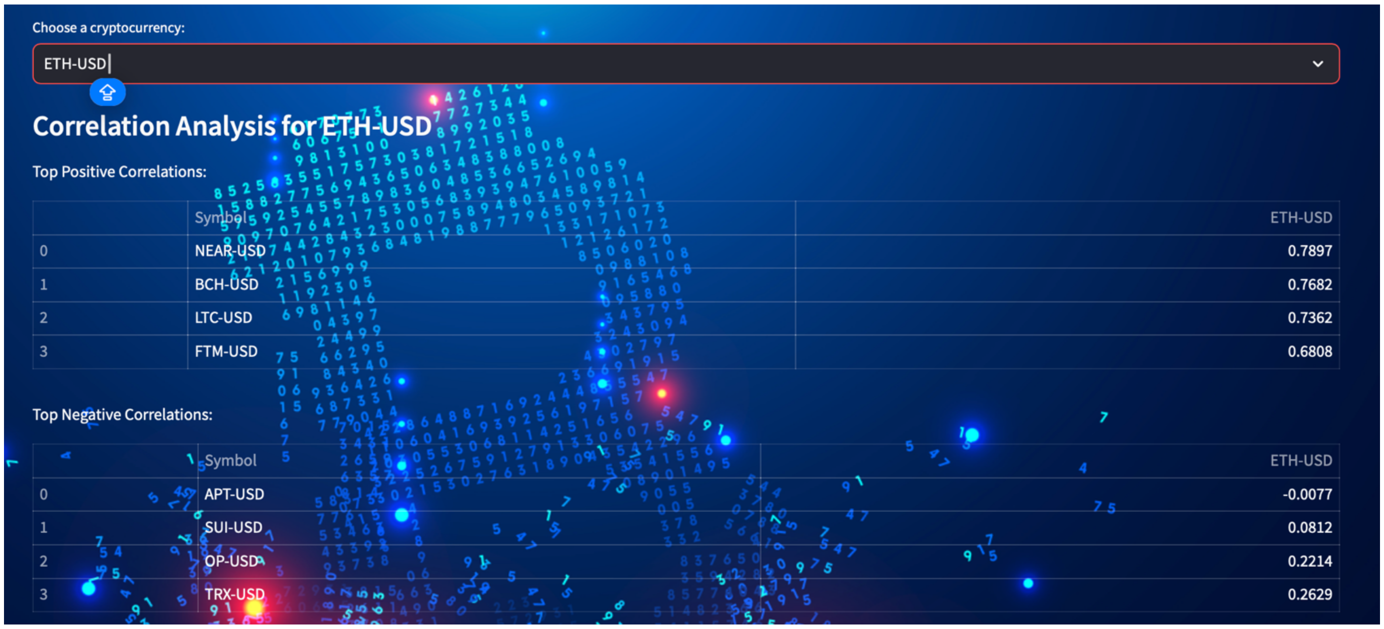


Figure 22 Historical Data viewer For BTC-USD



Choose a cryptocurrency:

ETH-USD

Correlation Analysis for ETH-USD

Top Positive Correlations:

	Symbol	ETH-USD
0	NEAR-USD	0.7897
1	BCH-USD	0.7682
2	LTC-USD	0.7362
3	FTM-USD	0.6808

Top Negative Correlations:

	Symbol	ETH-USD
0	APT-USD	-0.0077
1	SUI-USD	0.0812
2	OP-USD	0.2214
3	TRX-USD	0.2629

Figure 23 Top Positive and Negative Correlations For ETH-USD

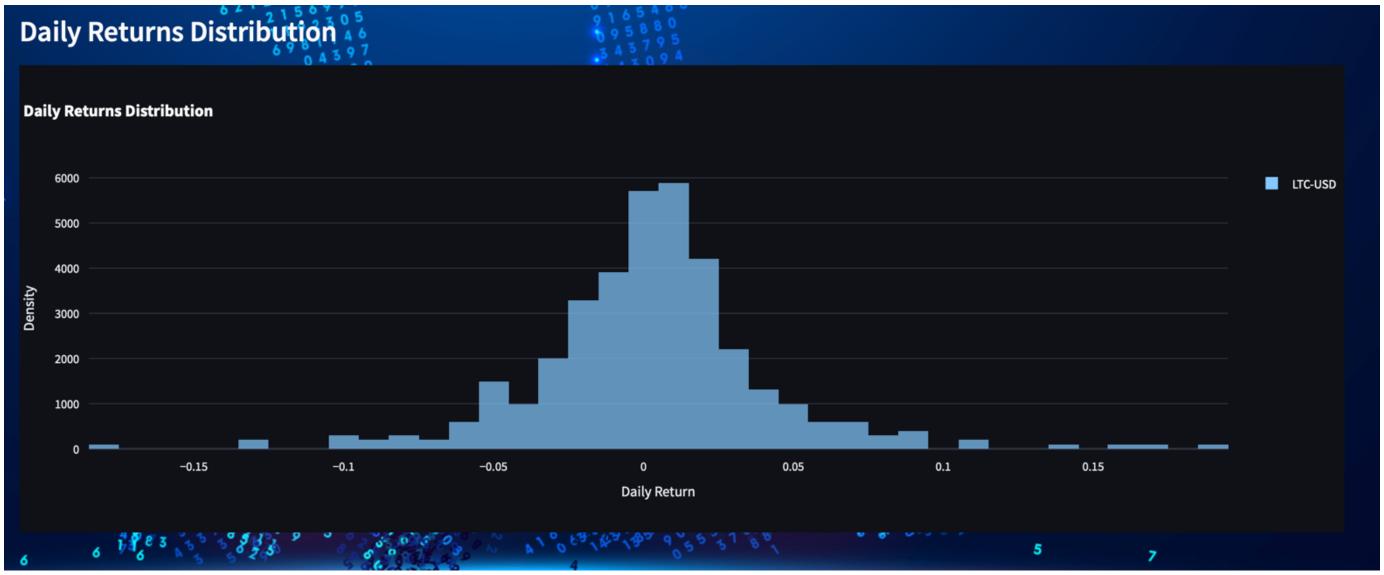
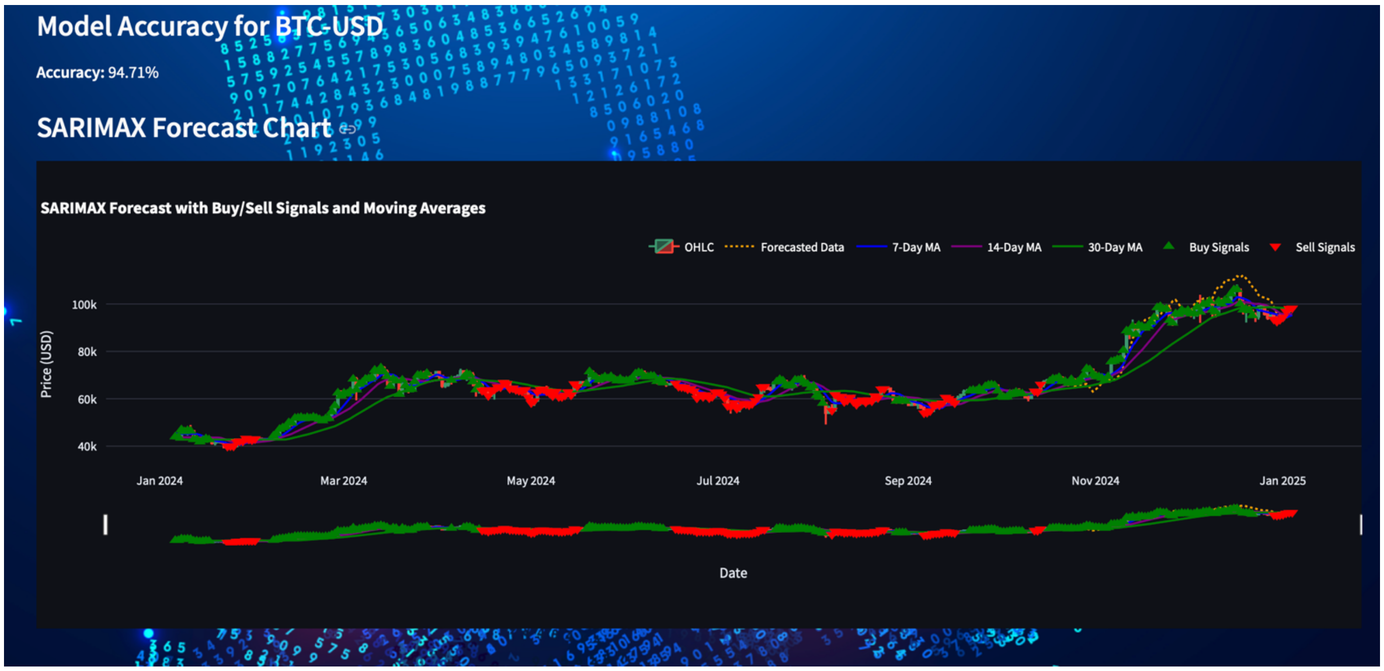


Figure 24 EDA ON LTC -USD



Model Accuracy for BTC-USD

Accuracy: 94.71%

SARIMAX Forecast Chart

Figure 25 Model prediction for SARIMAX Forecast

Limitation

The prototype showed promise but has limitations. It relies on real-time data, making it vulnerable to data dependency issues. Scalability challenges arise when managing large datasets or adding cryptocurrencies. The cryptocurrency market’s volatility affects long-term model accuracy. The intuitive GUI could be enhanced with additional customization options. These limitations offer opportunities for future improvements.

Conclusion

The ICT platform combines machine learning, visualization, and interactivity to address cryptocurrency trading challenges. It analyzes data, predicts trends, and provides insights, empowering informed trading decisions. SOLIGENCE leads fintech with a comprehensive, user-centric trading platform.

Reference List

Tungdajahirun, N., et al., 2023. Utilizing Artificial Intelligence in Cryptocurrency Trading: A Literature Review. *2023 7th International Conference on Information Technology (InCIT)*, Chiang Rai, Thailand, pp. 147-152.

Available at: https://ieeexplore.ieee.org/document/10413042

Seabe, P.L., Moutsinga, C.R.B. & Pindza, E., 2023. Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach. *Fractal and Fractional*, 7(2), p.203.
Available at: https://www.mdpi.com/2504-3110/7/2/203

Wu, J., Zhang, X., Huang, F., Zhou, H. & Chandra, R., 2024. Review of deep learning models for crypto price prediction: implementation and evaluation. *arXiv preprint*, arXiv:2405.11431v1 [cs.LG], 19 May.

Available at: https://arxiv.org/html/2405.11431v1

Yenidoğan, I., Çayir, A., Kozan, O., Dağ, T. & Arslan, Ç., 2018. Bitcoin forecasting using ARIMA and PROPHET. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, Bosnia and Herzegovina, pp. 621-624.

Available at: https://ieeexplore.ieee.org/document/8566476

Appendix I

Link to click and Download the Software: https://ssu-my.sharepoint.com/:f:/r/personal/0madaa90_solent_ac_uk/Documents/AE2_SOFTWARE_COM724?csf=1&web=1&e=XUPH9p