Script

Slide 1:[Introduction]

Hello Everyone. My topic for the presentation is the development of an Intelligent COin Trading Platform. The presentation is part of a task provided by Ultish Tech. So let's get into it.

Slide 2 & 3:[Contents & Problem Statement]

The contents of the presentation are as follows:
First, we'll go through the dataset we are provided by the company and discuss the parameter and importance of each.
We then start our discussion on Data preprocessing on our given dataset and analyze which pre-processing techniques are well suited and also the feature engineering process relevant to our models.
Third, we'll discuss our regression models, their suitability, advantages and limitation pertaining to our given task. Finally, we'll take into consideration the various evaluation metrics through which are models are scored on their accuracy and error rates.

Slide 4:[Dataset]

Given is the dataset for our cryptocurrency price prediction models. These are explained as follows:

1. id: The unique identifier for each cryptocurrency in the dataset.
2. name: The name of the cryptocurrency.
3. symbol: The symbol or ticker symbol representing the cryptocurrency.
4. slug: The URL-friendly version of the cryptocurrency name, often used in web applications and APIs.
5. num_market_pairs: The number of market pairs (trading pairs) available for the cryptocurrency.
6. date_added: The date when the cryptocurrency was added to the database or tracking platform.
7. max_supply: The maximum supply or total number of coins/tokens that will ever exist for the cryptocurrency.
8. circulating_supply: The current circulating supply of the cryptocurrency in the market.
9. total_supply: The total supply of the cryptocurrency, including both circulating supply and locked/uncirculated tokens.
10. platform_id: The unique identifier for the platform or blockchain on which the cryptocurrency operates, if applicable.
11. platform_name: The name of the platform or blockchain on which the cryptocurrency operates, if applicable.

12. platform_symbol: The symbol or ticker symbol representing the platform or blockchain, if applicable.
13. platform_slug: The URL-friendly version of the platform or blockchain name, often used in web applications and APIs, if applicable.
14. platform_token_address: The specific address or contract address of the cryptocurrency token on its respective platform or blockchain, if applicable.
15. cmc_rank: The rank of the cryptocurrency based on its market capitalization (market cap) compared to other cryptocurrencies.
16. last_updated: The date and time when the data for the cryptocurrency was last updated.
17. price: The current price of the cryptocurrency in a specified currency (e.g., USD, BTC, ETH).
18. volume_24h: The trading volume or total value of the cryptocurrency traded within the last 24 hours.
19. percent_change_1h: The percentage change in the price of the cryptocurrency over the last 1 hour.
20. percent_change_24h: The percentage change in the price of the cryptocurrency over the last 24 hours.
21. percent_change_7d: The percentage change in the price of the cryptocurrency over the last 7 days.
22. percent_change_30d: The percentage change in the price of the cryptocurrency over the last 30 days.
23. percent_change_60d: The percentage change in the price of the cryptocurrency over the last 60 days.
24. percent_change_90d: The percentage change in the price of the cryptocurrency over the last 90 days.
25. market_cap: The market capitalization of the cryptocurrency, calculated as the price multiplied by the circulating supply.
26. extracted_time: The date and time when the data was extracted or obtained from the data source.
27. count: The count or number of instances or data points for the specific cryptocurrency in the dataset.

Slide 5:[Data Preprocessing]

Data preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.
To optimize and clean the given dataset for cryptocurrency price prediction, the following data preprocessing steps can be performed:

1. Handle Missing Data:
    - Identify missing values: Check for missing values in each column of the dataset.
    - Decide on handling strategy: Determine the appropriate approach for handling missing data. Options include removing rows with missing values, imputing

missing values using techniques like mean, median, or forward/backward filling, or using more advanced imputation methods if appropriate.

2. Remove Irrelevant Columns:
    - Remove irrelevant features: Based on the relevance analysis mentioned earlier, remove columns such as ID, Slug, Num_market_pairs, Max_supply, Platform-related features (platform_id, platform_name, platform_symbol, platform_slug, platform_token_address), Extracted_time, and Count if they are considered irrelevant for price prediction.
3. Clean Up the Dataset:
    - Remove duplicates: Check for and remove any duplicate rows in the dataset.
    - Handle outliers: Identify outliers in numeric columns such as *Price, Volume_24h*, and *Market_cap*. Decide whether to remove outliers or apply appropriate transformations to handle them based on the specific characteristics of the data and the chosen machine learning model.
    - Check for inconsistent data types: Ensure that the data types of each column are appropriate. For example, numeric columns should be of numeric data type, and categorical columns should be of object/string data type.
4. Data Normalization:
    - Normalize numerical features: Normalize numeric columns like *Market_cap, Volume_24h, Circulating_supply, and Total_supply* to ensure they are on a similar scale. Common normalization techniques include **Min-Max scaling or Standardization**.
5. Categorical Variable Encoding:
    - Encode categorical variables: If there are categorical features like Name or Symbol, encode them using techniques like one-hot encoding or label encoding to convert them into numerical values that can be processed by machine learning algorithms.

Slide 6:[Feature Engineering]

Feature engineering is the pre-processing step of machine learning, which is used to transform raw data into features that can be used for creating a predictive model using Machine learning or statistical Modelling. Feature engineering in machine learning aims to improve the performance of models.

To optimize the given dataset for a machine learning model, the following feature engineering methods can be considered:

1. Date-based Features: From the *'date_added' and 'last_updated*' columns, extract useful features such as the day of the week, month, or year. These features can capture potential seasonality or periodic patterns in cryptocurrency prices.

2. Historical Price Features: We create new features that represent the percentage change in price over different time intervals (e.g., 1 hour, 24 hours, 7 days, etc.). These features can capture short-term price trends and volatility.
3. Rolling Statistics: Calculate rolling averages of price and volume over a specific time window (e.g., 7 days, 30 days). This can capture trends and smooth out short-term fluctuations.
4. Lagged Features: We can also introduce lagged versions of the target variable (price) or other relevant features. For example, you can create features representing the price from previous time steps (e.g., the previous day's price) to capture dependencies and momentum effects.
5. Seasonality and Holiday Effects: This is one of the very useful feature engineering methods yet not much used. If there are known seasonal or holiday effects on cryptocurrency prices, create features that capture such patterns. For example, if there is a historical trend of increased trading activity or price movement during certain holidays or events, include binary or categorical features to represent those periods.

We should remember to assess the relevance and impact of these engineered features through exploratory data analysis. Additionally, it's important to consider the limitations of historical data and the potential for changing market dynamics when incorporating engineered features. The prime example would be the change in prices seen in DOGE-COIN whenever the owner (Elon Musk) tweets about it.

Slide 7:[Machine Learning Models]

Coming to the core of our prediction system, which is choosing the best four models for predicting accurate cryptocurrency prices for future dates. I've shortlisted the following four models based on various parameters that we're going to discuss.

Slide 8:[Random Forest Regressor]

Random Forest algorithm combines ensemble learning methods with the decision tree framework to create multiple randomly drawn decision trees from the data, averaging the results to output a new result that often leads to strong predictions
Random Forest Regressor is a suitable model for the given dataset due to several reasons:

1. Handling Non-linearity: Cryptocurrency price prediction often involves complex patterns and non-linear relationships between features and the target variable. Random Forest Regressor can handle non-linear relationships effectively, making it suitable for capturing the intricate dynamics of cryptocurrency prices.
2. Robustness to Outliers: Cryptocurrency markets can be highly volatile, leading to outliers in the data. Random Forest Regressor is robust to outliers as it uses an ensemble of decision trees, which reduces the impact of individual outliers on the overall model.

3. Feature Importance: Random Forest Regressor provides a measure of feature importance. This helps in understanding which features have a stronger influence on the predicted prices, enabling better insights into the cryptocurrency market dynamics.
4. Handling High-Dimensional Data: The dataset may contain a large number of features, including historical price data, market-related information, and derived features. Random Forest Regressors can handle high-dimensional data efficiently without requiring extensive feature selection or dimensionality reduction techniques.

Algorithm of Random Forest Regressor: The Random Forest Regressor algorithm follows these steps:

1. Random sampling: Randomly select a subset of the training data (with replacement) to create a subset called a bootstrap sample.
2. Building Decision Trees: Construct multiple decision trees using the bootstrap sample. Each tree is built by recursively partitioning the data based on feature splits that maximize the information gain or minimize the impurity. However, at each split, only a random subset of features is considered.
3. Ensemble Voting: For each data point, predict the target value using each decision tree. The final prediction is determined by taking the average (regression) or majority vote (classification) of the predictions from all the decision trees.
4. Handling Overfitting: Random Forest Regressor mitigates overfitting by introducing randomness in two ways: (a) random sampling of the training data and (b) random selection of features at each split. This reduces the correlation between the trees and improves the generalization ability of the model.

Examples of Random Forest Regressor Applications: Random Forest Regressor has been widely used in various domains, including finance and stock market prediction, which share similarities with cryptocurrency price prediction.
Limitations of this model can include overfitting on the dataset provided along with being computationally expensive compared to other models.

Slide 9:[Extra Trees Regressor]

Similar to Random Forests, ExtraTrees is an ensemble ML approach that trains numerous decision trees and aggregates the results from the group of decision trees to output a prediction. However, there are a few differences between Extra Trees and Random Forest.
Random Forest uses bagging to select different variations of the training data to ensure decision trees are sufficiently different. However, Extra Trees uses the entire dataset to train decision trees. As such, to ensure sufficient differences between individual decision trees, it randomly selects the values at which to split a feature and create child nodes.
It also uses the entire dataset (which is the default setting and can be changed) allowing ExtraTrees to reduce the bias of the model. However, the randomization of the feature value at which to split increases the bias and variance.

In terms of computational cost, Extra Trees is much faster than Random Forest. This is because Extra Trees randomly selects the value at which to split features, instead of the greedy algorithm used in Random Forest.

Examples of Extra Trees Regressor Applications: The Extra Trees Regressor has been successfully applied in various domains, including finance and predictive analytics. Here are a few examples:

1. Financial Time Series Analysis: Extra Trees Regressor has been used to predict stock prices, exchange rates, and commodity prices based on historical price data and other relevant features.
2. Customer Churn Prediction: The model has been employed to predict customer churn in subscription-based services by analyzing customer behaviour, usage patterns, and demographic data.
3. Credit Risk Assessment: Extra Trees Regressor has been utilized to predict credit risk by analyzing customer financial data, credit history, and other relevant factors.
4. Demand Forecasting: The model has been applied to forecast demand for products or services based on historical sales data, market trends, and external factors.

Limitations of Extra Trees**:**

1. High Variance: Extra Trees models tend to have higher variance compared to Random Forest models due to the additional randomness introduced during feature selection and split point selection.
2. Feature Importance Estimation: Unlike Random Forest, Extra Trees do not provide built-in feature importance estimation. While feature selection is incorporated during the model training, obtaining explicit feature importance scores from Extra Trees requires additional analysis or techniques.

Slide 10:[Facebook's Prophet Model]

Prophet is an open-source tool from Facebook used for forecasting time series data which helps businesses understand and possibly predict the market. It is based on a decomposable additive model where non-linear trends fit with seasonality, it also takes into account the effects of holidays.
Prophet is specifically designed for time series forecasting, making it well-suited for predicting future cryptocurrency prices based on historical patterns.
Cryptocurrency markets often exhibit seasonal patterns, periodic fluctuations, and long-term trends. Prophet incorporates mechanisms to capture such seasonality and trends, allowing it to model and predict these patterns accurately. Prophet is also designed to be user-friendly and accessible, even for users without extensive expertise in time series forecasting.

Algorithm of Prophet Model: The Prophet model utilizes the following algorithm:

1. Trend Estimation: Prophet begins by estimating the underlying trend in the time series data. It fits a piecewise linear or logistic function to capture the overall trend, including any abrupt changes or transitions.

2. Seasonality Modeling: Prophet incorporates seasonality patterns, such as weekly, monthly, or yearly cycles, based on historical data.
3. Holiday Effects: The model allows for the inclusion of custom or pre-defined holiday effects that have a significant impact on the time series data. This is particularly useful in cryptocurrency analysis, as certain holidays or events can influence market behaviour.
4. Bayesian Forecasting: Prophet applies a Bayesian framework to generate future predictions. It constructs a range of potential future trajectories by simulating variations in trend, seasonality, and holiday effects. This approach enables the quantification of prediction intervals and uncertainty estimation.

Examples of Prophet Model Applications: The Prophet model has been applied in various domains for time series forecasting.

Limitations of Facebook's Prophet:

1. Linearity Assumption: Prophet assumes that the underlying trends and seasonality follow linear patterns. It may not capture complex nonlinear relationships in the data.
2. Limited Exogenous Factors: Prophet focuses primarily on modelling the time series components (trend, seasonality) and may not handle external factors well. It may not be suitable for scenarios where the target variable is influenced by a significant number of external factors beyond the provided data.

Slide 11: [Lasso Regression Model]

Lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting model.
The Lasso Regressor model is suitable for the above dataset for the following reasons:

1. Feature Selection: The Lasso Regressor performs both regularization and feature selection simultaneously. effectively selecting a subset of the most relevant features for predicting cryptocurrency prices. This is beneficial when dealing with high-dimensional datasets like cryptocurrency data, where not all features may contribute significantly to the target variable.
2. Handling Multicollinearity: Cryptocurrency datasets may contain highly correlated features, which can lead to multicollinearity issues. The Lasso Regressor can handle multicollinearity by penalizing the coefficients of correlated features.

Algorithm of Lasso Regressor: The Lasso Regressor algorithm follows these steps:

1. Initialization: Initialize the coefficients (weights) of the model with small random values.
2. Loss Function: Define a loss function that measures the difference between the predicted values and the actual values of the target variable. The loss function in Lasso Regressor consists of two terms: the sum of squared errors (similar to ordinary least squares) and the L1 norm of the coefficients multiplied by a regularization parameter (alpha).

3. Minimization: Minimize the loss function by adjusting the coefficients using an optimization algorithm such as coordinate descent or gradient descent.
4. Regularization: The Lasso Regressor introduces an L1 regularization term, which penalizes the absolute values of the coefficients.
5. Feature Selection: The Lasso Regressor automatically selects a subset of features by setting the coefficients of irrelevant features to zero.

Examples of Lasso Regressor Applications: The Lasso Regressor has been successfully applied in various domains, including finance, economics, and healthcare. Here are a few examples:

Limitations of Lasso Regressor:

1. Feature Selection Bias: Lasso Regressor tends to select features based on their individual predictive power and may overlook important feature interactions.
2. Deterministic Feature Selection: Lasso Regressor consistently selects the same features for a given dataset and regularization parameter. This can be a limitation if the dataset characteristics change or if different subsets of features are relevant under different circumstances.

Slide 12:[Evaluation Metrics]

Now to measure how well our model fared on the dataset it's important to measure them using some predefined metrics. The four metrics which are most relevant to us are:

1. R-Squared(R^2 Score): R-squared measures the proportion of variance in the target variable that is explained by the model. It ranges from 0 to 1, where 1 indicates a perfect fit and 0 indicates that the model does not explain any of the variances. Higher R-squared values indicate better performance.
2. Mean Squared Error(MSE): MSE calculates the average squared difference between the predicted values and the actual values. It gives a measure of the average magnitude of the error. Lower MSE values indicate better performance.
3. Root Mean Squared Error (RMSE): RMSE is the square root of MSE and provides a more interpretable metric in the same unit as the target variable. It gives a measure of the average magnitude of the error with the same scale as the target variable.
4. Mean Absolute Error (MAE): MAE calculates the average absolute difference between the predicted values and the actual values. It provides a measure of the average magnitude of the errors without considering their direction. Lower MAE values indicate better performance.

Slide 13:[Conclusion]

In conclusion, we have discussed the development of an Intelligent Coin Trading Platform, focusing on the dataset provided by Ultish Tech. We explored the importance of handling missing data, removing irrelevant columns, cleaning up the dataset, normalizing numerical features, and encoding categorical variables.

Furthermore, we discussed four machine learning models suitable for predicting cryptocurrency prices: Random Forest Regressor, Extra Trees Regressor, Facebook's Prophet Model, and Lasso Regression Model. We highlighted the advantages and limitations of each model and provided examples of their applications in various domains.

The Random Forest Regressor is well-suited for handling non-linearity, robustness to outliers, and high-dimensional data. The Extra Trees Regressor offers faster computation and has been successfully applied in finance and predictive analytics. Facebook's Prophet Model is designed specifically for time series forecasting, capturing seasonality and trends, while Lasso Regression Model performs variable selection and regularization, making it suitable for high-dimensional datasets.

It is important to note that each model has its own limitations, such as overfitting in the Random Forest Regressor, higher variance in the Extra Trees Regressor, linearity assumption in Facebook's Prophet Model, and limited handling of external factors in the Lasso Regression Model.

In conclusion, the choice of the best model for the Intelligent Coin Trading Platform depends on the specific requirements and characteristics of the dataset. Careful consideration should be given to the features, data patterns, computational resources, and interpretability needed for accurate cryptocurrency price prediction.