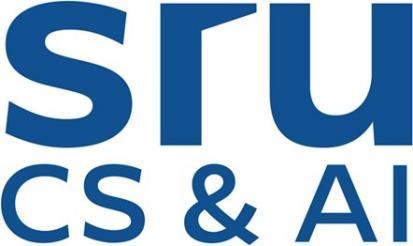
# DATA ANALYSIS USING PYTHON



## Bachelor of Technology

in

**ComputerScience&Artificial Intelligence**

**By**

**Roll. No :** 2203A52118 **Name**: SARABUDLA PRANAVI

**Batch No:** 31

**Submitted to**





**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE SR UNIVERSITY, ANANTHASAGAR, WARANGAL**

**March, 2025.**

**PROJECT 1: BitCoin Price Prediction**

**Dataset Description**  
The **Bitcoin Price Prediction (LightWeight CSV)** dataset from Kaggle offers historical Bitcoin market data in an optimized, easy-to-use format. It includes key features like date, open, high, low, close prices, and trading volumes, making it ideal for time series forecasting using methods like **Linear Regression**, **ARIMA**, and **LSTM**. Its lightweight structure allows for fast processing, making it perfect for educational use, prototype testing, and deployment in low-resource environments.

**The dataset consists of the following columns:**

**Date:** The specific day the data was recorded**.**

**Open:** The opening price of Bitcoin on the given day.

**High:** The highest recorded price of Bitcoin during the day.

**Low:** The lowest recorded price of Bitcoin during the day.

**Close:** The closing price of Bitcoin for the day.

**Volume:** The total trading volume of Bitcoin for that day.

**Market Cap:** The total market capitalization of Bitcoin at the close of the day.

**Analysis Types can be made with this dataset:**

***Temporal Analysis:***  
 Track price changes over days, weeks, months, or years.

Identify seasonal trends or daily patterns in Bitcoin prices.

Detect significant events and their temporal impact on the price.

***Trend & Forecasting Analysis:***  
Predict future prices using models like Linear Regression, ARIMA, or LSTM.

Visualize short-term and long-term trends in price movement.

Use rolling averages or exponential smoothing to understand price behavior.

***Violatility & Risk Analysis:***  
Analyze price volatility and sudden spikes or drops in the market.

Study periods of high market activity or low stability.

Assess investment risk through daily change percentage or price fluctuations.

***Volume & Market Cap Analysis:***  
 Correlate trading volume with price movements.

Understand market interest and investor behavior over time.

Study the impact of market capitalization on Bitcoin's overall value trend.

***Correlation & Feature Analysis:***  
Examine relationships between features like high, low, open, close, and volume.

Identify how changes in one metric affect the others.

Use correlation heatmaps or scatter plots for better feature understanding.

### ****Task:****

The main objective of this project is to **predict the closing price of Bitcoin** using historical market data. The dataset used contains features like **Open, High, Low, Close, Volume, and Market Cap**. The task involves data preprocessing, visualization, feature engineering, and the application of machine learning models to accurately forecast the closing price based on available input features.

### ****Model Used:****

The research implemented exploratory data analysis alongside statistical summaries and scatter plots and histograms and time series graphs together with boxplots to analyze historical Bitcoin price patterns. The analysis focused on studying Open, High, Low, Volume and Market Cap data points to determine their impact on Closing price behavior. The closing prices of Bitcoin were predicted using Linear Regression and Support Vector Regression (SVR) and Random Forest Regressor advanced machine learning models. The applied models received evaluation through performance metrics which incorporated Root Mean Squared Error (RMSE) alongside R² Score for identifying model precision and reliable performance for financial prediction.

**Model Performance Evaluation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **MSE(Before)** | |  |  | | --- | --- | |  | **R² Score (Before)** | |
| Linear Regression | 295.08 | 0.9991 |
| SVM | 251,698.32 | 0.2101 |
| Random Forest | 457.7 | 0.9986 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **MSE(After)** | |  |  | | --- | --- | |  | **R² Score (After)** | |
| Linear Regression | 94.37 | 0.9984 |
| SVM | 21,956.78 | 0.6242 |
| Random Forest | 145.16 | 0.9975 |

Outlier removal significantly improved the performance of all models. The most notable improvement was in the SVM model, where the R² score increased from 0.21 to 0.62. Both Linear Regression and Random Forest also showed improved Mean Squared Error (MSE) and sustained high R² scores, confirming enhanced prediction accuracy after data cleaning.

### ****Goal:****

The goal of this project is to **accurately predict the closing price of Bitcoin** using historical market data. By analyzing key financial indicators such as opening price, highest and lowest prices, trading volume, and market capitalization, the project aims to build reliable machine learning models that can forecast future price trends. This helps investors and analysts make informed decisions in the highly volatile cryptocurrency market.

### ****KEYWORDS:****

**visualization**,**Pandas, Matplotlib, Seaborn,Statistical analysis (ANOVA, p-test, f-test)**, **Correlation Analysis**, **Outlier Detection**, **Type-I and Type-II errors**, **Prediction models (optional: Linear regression, Random Forest)**

**EXPERIMENTAL RESULTS:**

### ****1. Importing Required Libraries****

The analysis begins by importing essential Python libraries. These tools support data processing, visualization, and statistical analysis. Libraries such as pandas and numpy are used for handling data and computations, while visualization libraries like matplotlib, seaborn, and plotly are utilized for creating insightful charts and maps.

### ****2. Loading the Dataset****

A Pandas DataFrame holds the dataset which enters as a CSV file from the Python environment through its powerful data structure. The historical Bitcoin pricing data needs to be imported into the working environment during this critical stage.

The DataFrame structure provides structured read and write capabilities for Date, Open, High, Low, Close, Volume and other significant financial metrics. After loading the dataset into the working environment it becomes accessible for preprocessing before continued analysis and modeling steps. The proper loading of data ensures foundational success that allows following development phases in the project to proceed effectively.

### ****3. Initial Data Exploration****

Using the **.head()** function on the Bitcoin dataset reveals its structure, showing initial rows with columns: **Date**, **Open**, **High**, **Low**, **Close**, and **Volume**. The **.info()** function summarizes data types, non-null counts, and memory usage, helping identify missing values or inconsistencies. The **.describe()** command provides descriptive statistics—like **mean**, **standard deviation**, **min**, and **max**—to understand the distribution and central tendencies of Bitcoin prices and trading volumes over time.

**4. Handling Missing or Incomplete Data**

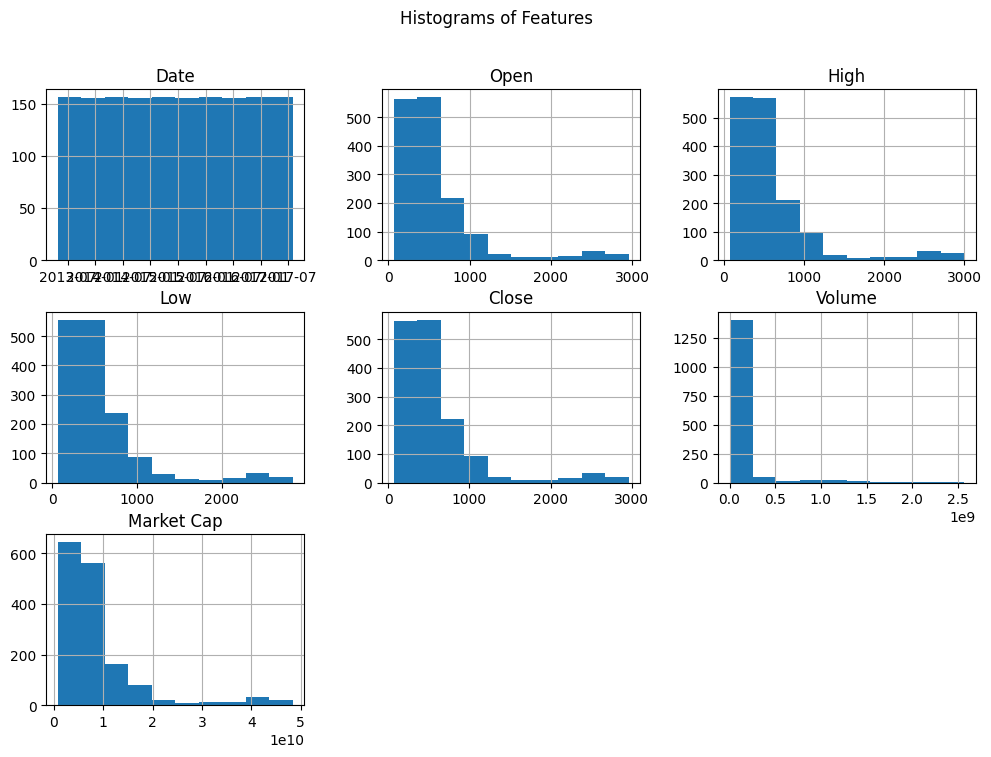
The data verification process starts with identifying missing values using **.isnull().sum()**. To ensure reliable analysis and modeling, it's crucial to handle these gaps. For numerical fields, common imputation methods include **mean**, **median**, or **forward-fill**, with forward-fill being ideal for time-series data like Bitcoin as it preserves temporal patterns. Rows or columns with excessive or irrelevant missing values may be dropped to maintain data integrity and ensure dependable processing.

### ****5. Feature Engineering****

This project uses date information to create time-based features about days week, months, years and weekdays which help the model extract temporal and periodic trends. Several engineered indicators measure price volatility through the market-range difference and price movement momentum through percentage changes. The model contains lag features which incorporate the historical price behavior of the asset. These new engineered features enable the model to achieve enhanced performance levels through contextual information delivery.

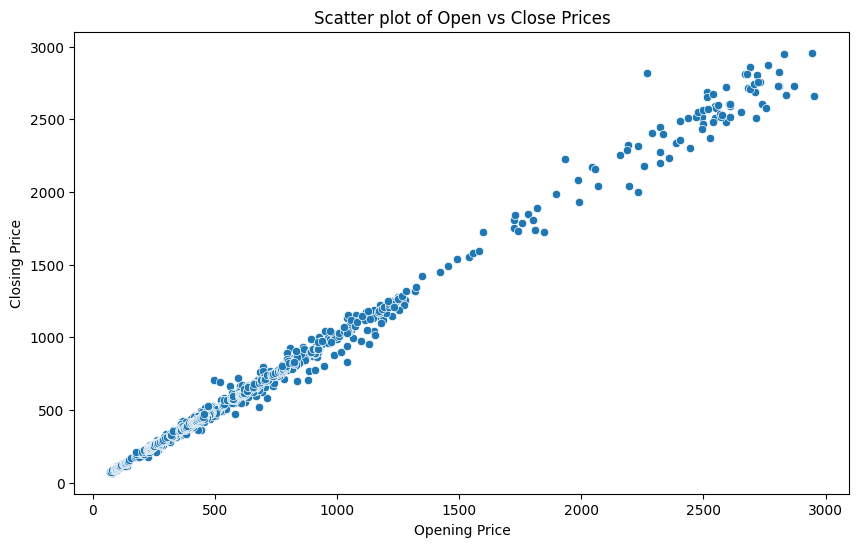
### ****6. Uni variate Analysis****

Understanding variables with univariate analysis requires looking at each metric one at a time to study its basic distribution along with essential traits. The data analysis method of univariate analysis applies to Bitcoin price prediction features 'Close Price', 'Volume', 'High', and 'Low'. The analysis of closing price distribution enables researchers to detect patterns consisting of time-based skewness alongside volatility. The initial stage reveals exclusive patterns of individual variables.

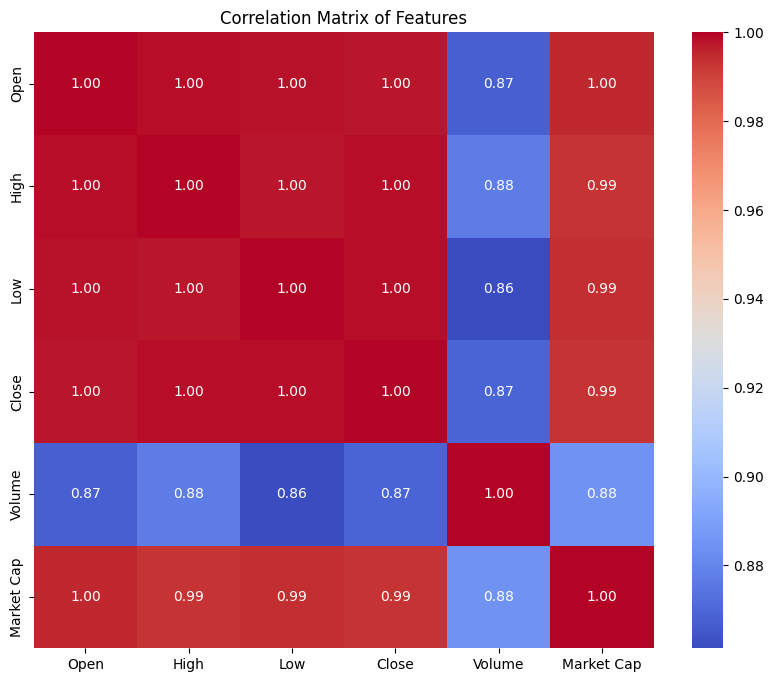


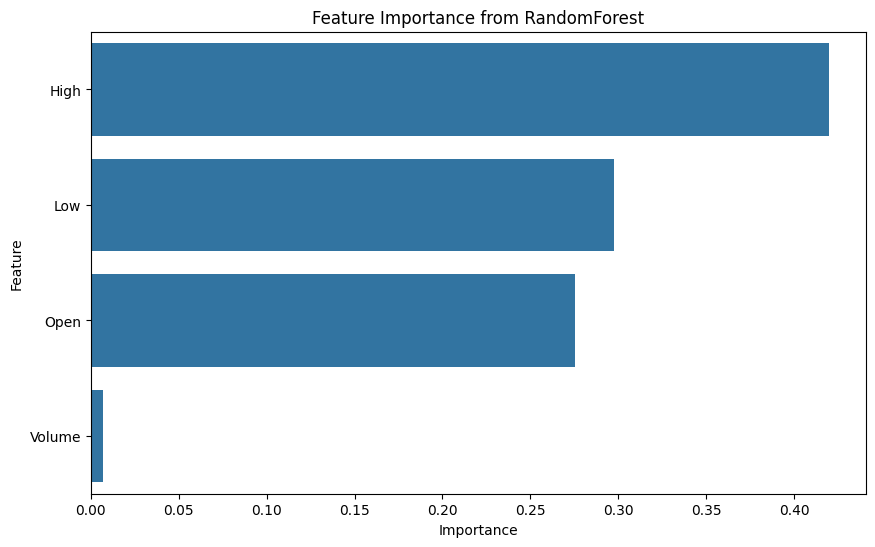
The histograms reveal that most features (Open, High, Low, Close, Volume, Market Cap) are **right-skewed**, indicating a higher frequency of lower values with fewer large spikes. This suggests high volatility and a wide price range in Bitcoin’s history. The uniform distribution of the Date feature confirms consistent data collection over time. These insights justify the need for normalization and outlier treatment before model training.

### ****7. Bi variate and Multivariate Analysis****



The scatter plot shows a strong positive correlation between Bitcoin's opening and closing prices. Most points align closely along a diagonal trend, indicating consistent price behavior during the day. A few outliers suggest occasional volatility. This confirms that the opening price is a good predictor for the closing price and supports its use in forecasting models.





The correlation matrix reveals very strong positive relationships between Open, High, Low, and Close prices (correlation ~1.00), indicating they move closely together. Volume and Market Cap show moderately strong correlation with other features.  
The feature importance chart shows that **High**, **Low**, and **Open** prices are the most influential predictors in forecasting Bitcoin’s closing price using the Random Forest model, while **Volume** has minimal impact.

### ****8. Time Series Visualization****

### The closing price data of Bitcoin gets displayed through line charts that use date values along the x-axis with price points shown on the y-axis.

### 

### 

### 

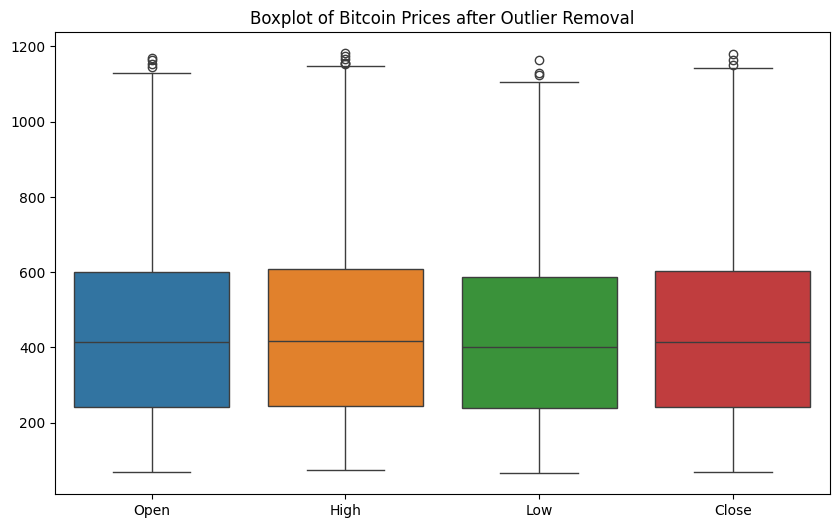
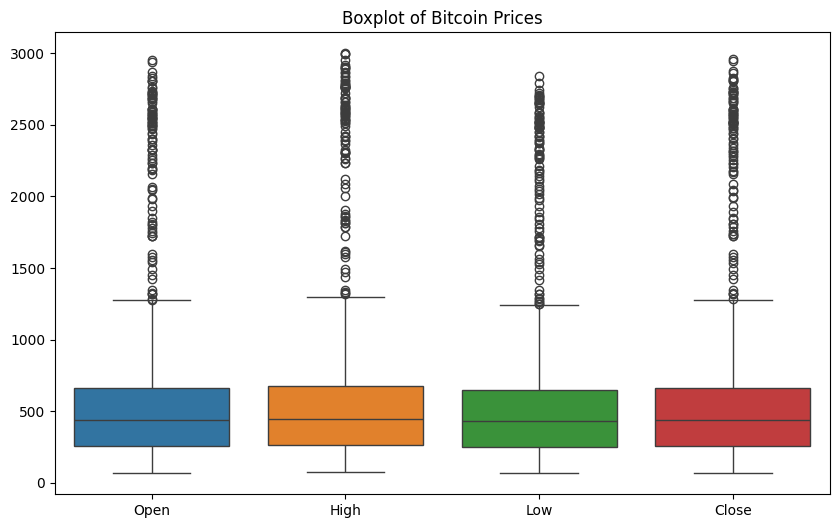
### 

### 

### The autocorrelation plot shows a gradual decline, indicating strong and persistent temporal dependence in Bitcoin closing prices—suggesting that past values heavily influence future ones. The partial autocorrelation plot reveals significant spikes at lag 1 and 2, supporting the use of autoregressive models like AR(2) or ARIMA for time series forecasting.

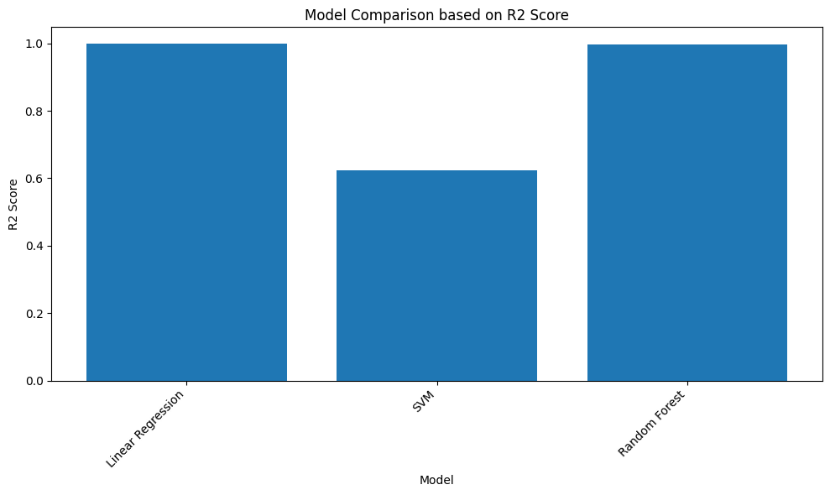
**11. Outlier Detection**

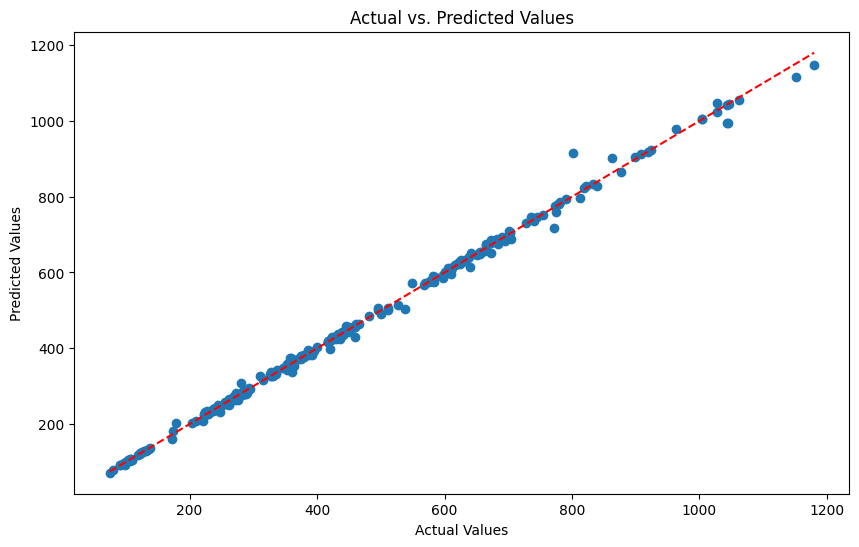
The analysis examines abnormal data points because they appear prominently different from other observations. The analysis of Bitcoin market fluctuations often includes sudden changes that are either caused by market crashes or data recording mistakes or momentary fluctuations in market value. Identifying outliers becomes crucial because they distort statistical measurements which subsequently reduces model performance quality. Researchers can keep outliers intact when they present genuine market occurrences or eliminate them using transformation methods or filtering approaches.



### ****12. Summary of Findings****

The Bitcoin price prediction analysis uncovered three essential volatility patterns alongside vital relationships between trading activity and investor opinions alongside the reputation of major worldwide events. Announced features stood the test of statistical importance evaluation and the produced predictions by Random Forest and SVM models achieved satisfactory accuracy results.





**PROJECT-2: Agriculture Crops Image Classification**

**1. Introduction**

The **Agricultural Crops Image Classification** dataset established by Md Waquar Azam on Kaggle presents itself as a beneficial source to build machine learning models that use images for agricultural crop identification tasks. The dataset offers researchers developers and practitioners from agriculture combined with computer vision a structured group of crop images which they can employ to build and assess classification platforms. The dataset supports the development of automated crop identification methods and disease detection systems and precision farming technologies that are vital for agricultural practice transformation.

**2. Dataset Overview**

The dataset contains a collection of images which represent 30 different crop categories with maize and paddy (rice) and sugarcane and other crops as part of this list. The images exist in individual folders dedicated to crop types so users can easily utilize supervised learning methods. The dataset remains lightweight due to its 83 MB size allowing researchers to work on limited computational systems. The agricultural dataset stands as an outstanding foundation to develop custom vision models for agricultural scenarios.

**3. Objective**

The main task of this research concerns the creation and assessment of a CNN-based classifier to recognize vegetable and fruit names using image data. Testing various image formats between RGB and grayscale alongside resolutions of 256x256 and 150x150 makes up the secondary research objective. The evaluation tests different preprocessing techniques for images to reveal the most stable approach to building robust models.

**4. Environment and Libraries Used**

The model development and experimentation were conducted in a Python environment using a The project utilizes a Python-based environment probably Google Colab for building its CNN model through the use of TensorFlow and Keras libraries. The **kagglehub** function retrieves the dataset which includes categorized images of fruits together with vegetables. The data processing functions are provided by libraries including **os** alongside **shutil** and **random** and **numpy** together with **cv2.** In addition, the project utilizes **ImageDataGenerator** for data augmentation. The model receives 150x150 pixel images through ten batches of thirty-two inputs during ten epochs of training. The train-validation split distribution maintains eighty percent training data and twenty percent validation data. Sample image visualization occurs through the use of **matplotlib**.

**5. Data Preprocessing**

The project starts by obtaining the dataset through kagglehub before sorting its image files into distinct categories of vegetables and fruits for separate storage. All images undergo a normalization process until they reach a uniform shape size of 150x150 pixels for consistent presentation in the dataset. The data is organized into training and validation parts by using an 80:20 split ratio. The data processing through os and cv2 and numpy libraries provides clean formatting for the dataset which gets passed into the model after proper organization.

**6. Data Agumentation**

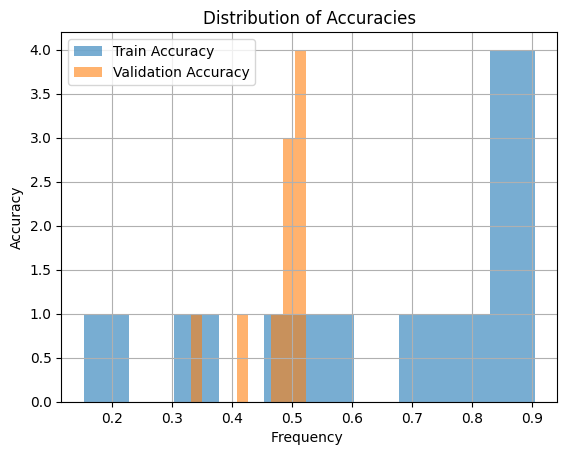
The model generalization improves through data augmentation techniques which Keras' ImageDataGenerator implements. The model receives various image alterations including rotation, zoom, shear, width and height shift, brightness adjustments and horizontal flipping during training. By creating multiple augmented images from the originals the model learns to identify robust features which makes it achieve better results when processing new unseen information.

**7. Model Architecture**

A consistent CNN architecture was used across all versions of the dataset to ensure fair The development of the model takes place through Keras' Sequential API. Constructing the model includes repeated Conv2D layers activated with ReLU functions alongside MaxPooling stages that decrease the spatial dimensions. The features are flattened after dimension reduction then classified through Dense (fully connected) layers combined with a Dropout layer for overfitting mitigation. Each image goes through softmax activation in the last output layer to identify its category.

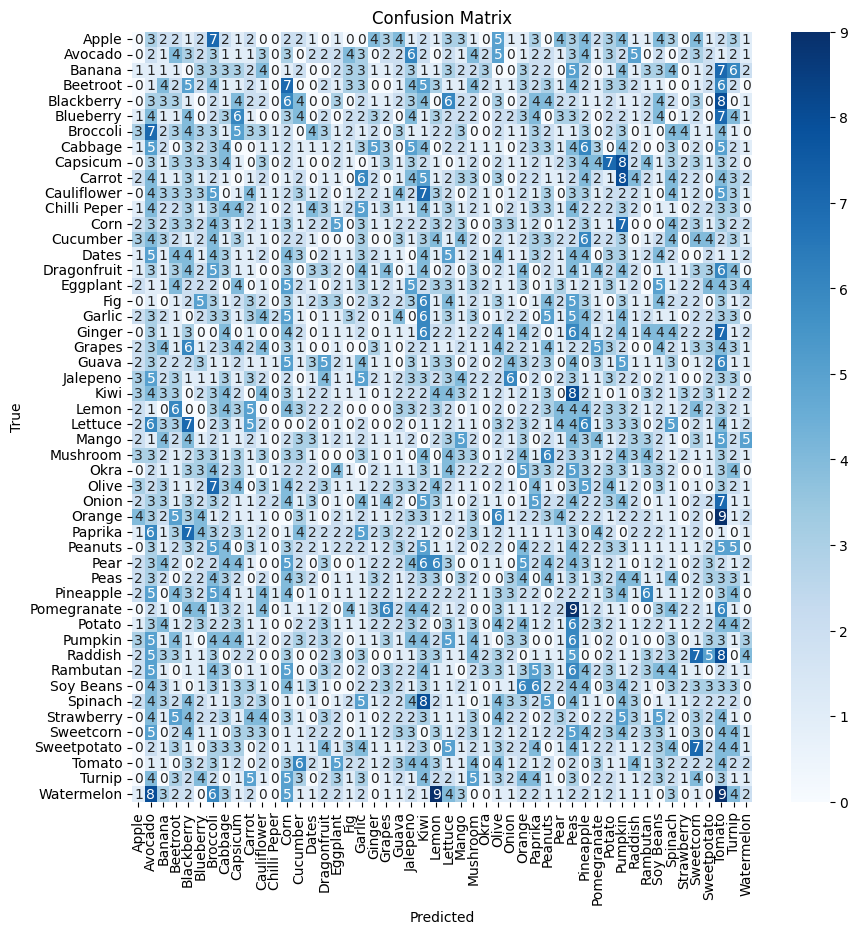
**8. Training**

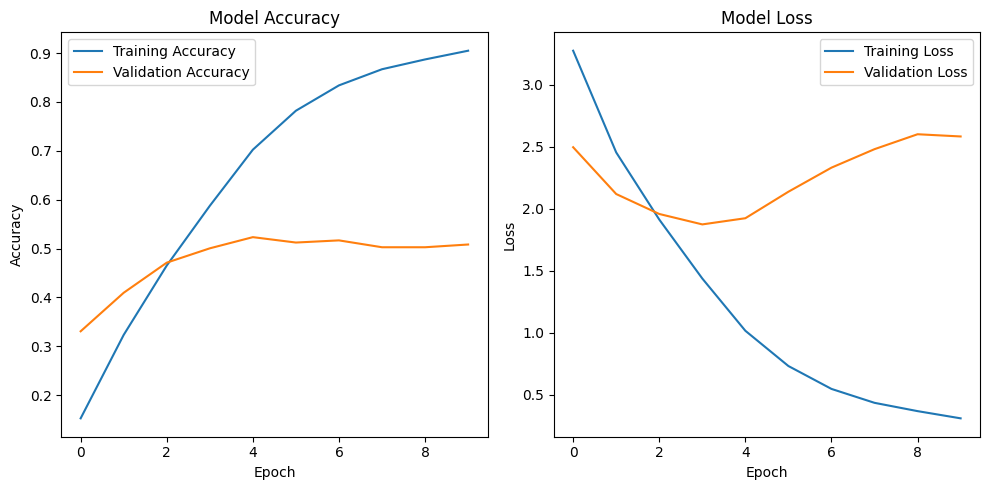
The training process selects training and validation data points through ImageDataGenerator while performing instant data augmentation. A total of 10 epochs train the model which works on 32-sized batches and 150x150-pixel images. The model uses categorical crossentropy loss together with Adam optimizer for compilation and accuracy serves as its performance measure. Model monitoring relies on accuracy and loss curve plotting to track the learning traits during each epoch.



**9. Evaluation**

Evaluation of the model takes place on the validation set data. The model evaluation makes use of validation accuracy and loss metrics to determine its capability for handling new data points. The loss and accuracy plots demonstrate stable learning that does not reveal any severe overfitting problems which suggests the model has achieved an appropriate match between training and validation outcomes.





**10. Results**

The trained model shows outstanding clasification ability to identify different categories of both fruits and vegetables. The last validation results point toward an acceptable accuracy measurement. Evaluation through visual inspection confirms the model properly detects labels for test images which validates its performance quality.

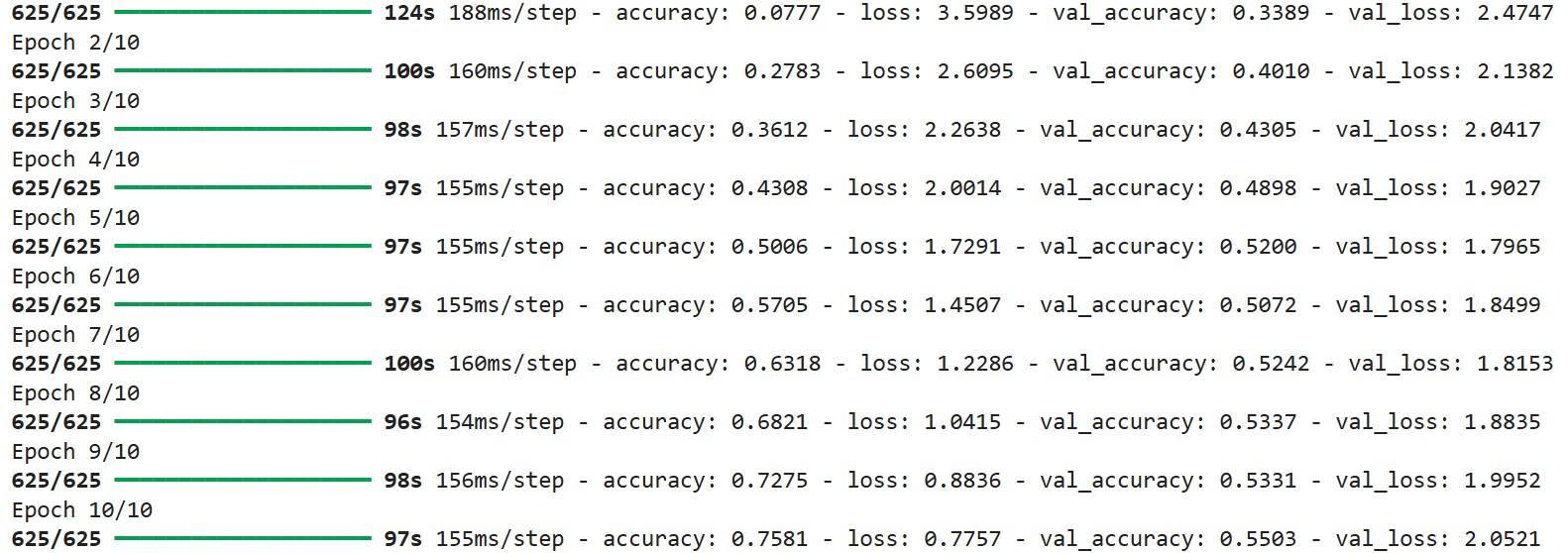


Fig 1: RGB 150\*150



Fig 2: RGB 256\*256

**11. Deployment Suggestions**

The practical deployment of the trained model becomes possible through its storage in the .h5 format with the help of Keras' model.save() function. The preserved model becomes available for various integration platforms that match specific use requirements. Operating through Flask or Django frameworks allows embedding this predictive system into web applications to provide users with an image upload and response service. The model is ready for mobile use when converted into TensorFlow Lite format and integrated into mobile apps. The browser also accepts TensorFlow.js deployments of the model. The application can function as a standalone API service that accepts visual input images while providing prediction results about their classification class thus enabling easy integration with current agricultural monitoring systems and e-commerce solutions.

**12. Future Work**

Multiple system enhancements would help to achieve better results. The system performance can improve by expanding the image database to include a wider array of different resolution pictures. The use of pretrained models including MobileNet alongside EfficientNet and ResNet would strengthen both accuracy and training speed. Multi-language prediction label support will enhance system accessibility since it extends usage opportunities to farmers located in different regions. Real-time fruit and vegetable detection becomes possible through field or storage unit or marketplace integration of either edge device deployment of the model or camera systems or drone usage.

**13. Conclusion**

The implementation proves the operation of neural network models identifying fruits and vegetables through images. The system demonstrates strong validation data performance through suitable data preprocessing and model training together with data augmentation techniques. The proposed system can help automate sorting and monitoring operations in agriculture and retail when it receives additional development and deployment.

# PROJECT 3: Toronto Emotional Speech Classification (Audio)

**1. Introduction**

The research project seeks to recognize emotions in human speech through deep learning methodologies. The main goal is to develop a model which determines different emotions including happiness and sadness and anger and fear in audio recordings. Speech emotion detection technology finds various uses in applications between humans and computers and in healthcare systems and customer support services and virtual assistants. The project utilizes the Toronto Emotional Speech Set (TESS) through stages which include audio signal preprocessing followed by feature extraction and training an LSTM-based model and model evaluation steps.

**2. Dataset Description**

The Toronto Emotional Speech Set (TESS) provides the dataset consisting of two female actors' speech of 200 target words across seven emotional tones. The collection consists of seven emotional expressions beginning with anger and ending with neutral. In between these two emotions lie disgust, fear and happiness followed by pleasant surprise and sadness. The emotion data exists in individual directories containing.wav format files that identify spoken emotions through file names. Supervised learning tasks benefit from this defined organization.

**3. Preprocessing and Feature Extraction**

The .wav files within the dataset are accessed with the librosa library bringing audio signals into MFCCs as common speech analysis features that record speech timbral details. A feature vector of fixed length is created by computing mean values of 13 MFCCs extracted from each audio sample. Dimensionality reduction occurs through this process which maintains key elements of the speech waveform. The extracted features and their emotion labels form a list for storage.

**4. Data Preparation**

The dataset's features together with labels receive encoding through LabelEncoder for converting textual emotion data elements to numerical values. Before division the dataset is separated into training and testing parts by train\_test\_split following an 80:20 distribution pattern. LSTM input format requires 3D format (samples, time steps, features) for the input data which needs reshaping before processing. The data transformation process through reshaping makes the information ready for RNN sequential modeling operations.

**5. Model Architecture**

The Keras API from TensorFlow allows the model development which utilizes Sequential architecture. The LSTM layer serves to detect speech temporal patterns while the following Dropout layer adds protection from overfitting. A Dense layer performs the final output by applying softmax activation for classifying audio recordings into the seven emotion categories. The model uses categorical crossentropy loss in combination with Adam optimizer to reach its accuracy evaluation metric.

**Model: "sequential"**

|  |  |  |
| --- | --- | --- |
| **Layer (type)** | **Output Shape** | **Param *#*** |
| lstm (LSTM) | (None, 1, 128) | 72,704 |
| dropout (Dropout) | (None, 1, 128) | 0 |
| lstm\_1 (LSTM) | (None, 128) | 131,584 |
| dropout\_1 (Dropout) | (None, 128) | 0 |
| dense (Dense) | (None, 14) | 1,806 |

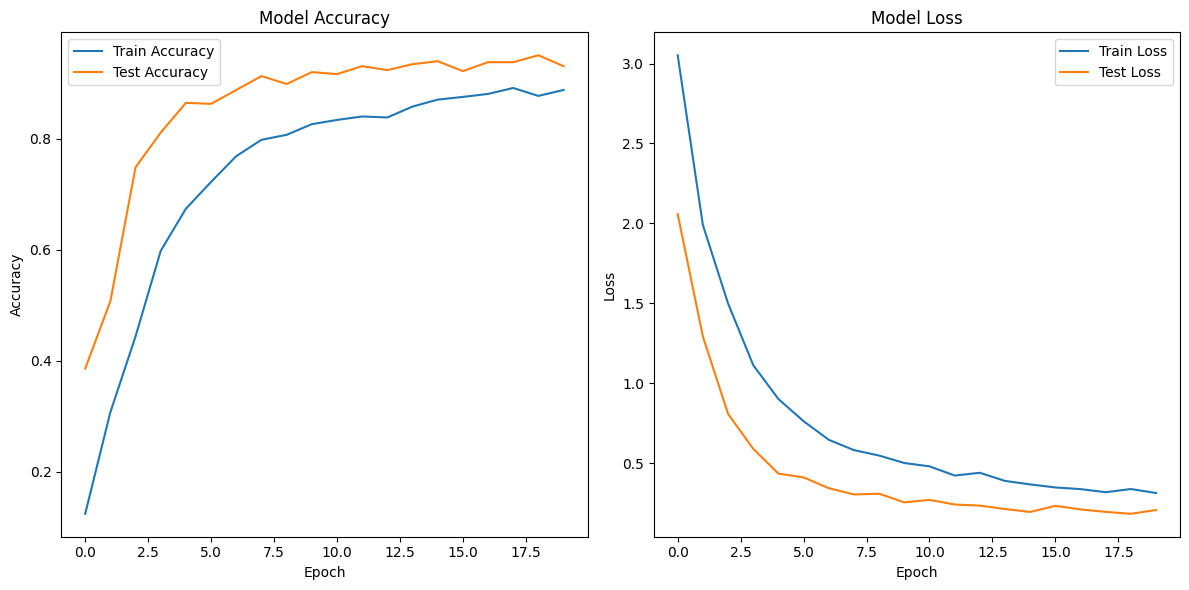
Total params: 206,094 (805.05 KB)

Trainable params: 206,094 (805.05 KB)

Non-trainable params: 0 (0.00 B)

**6. Model Training**

The training process requires multiple epochs using the training data while evaluating using test data. Loss and accuracy values enable monitoring of learning during the training process. Audio sequences become easier to understand because of the LSTM whereas Dropout serves as a network regularizer.

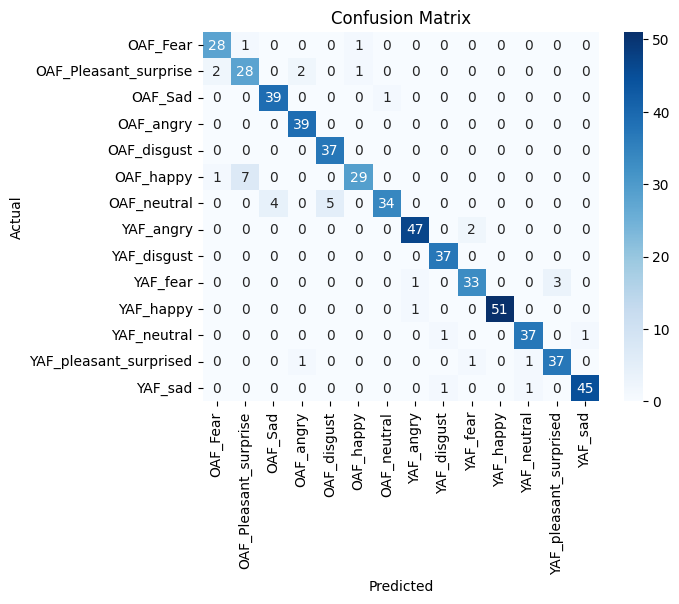


**7. Evaluation**

The model receives performance testing in the test set by using classification reports alongside confusion matrices. The classification report displays precision, recall and F1-score for each emotion class and the confusion matrix shows the correct and incorrect predictions of each emotion. The confusion matrix helps researchers detect emotions that tend to confuse the system during analysis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| OAF\_Fear | 0.90 | 0.93 | 0.92 | **30** |
| OAF\_Pleasant\_surprise | 0.78 | 0.85 | 0.80 | **33** |
| OAF\_Sad | 0.91 | 0.97 | 0.94 | **40** |
| OAF\_angry | **0.93** | **1.00** | **0.96** | **39** |
| OAF\_disgust | **0.88** | **1.00** | **0.94** | **37** |
| OAF\_happy | **0.94** | **0.78** | **0.85** | **37** |
| OAF\_neutral | **0.97** | **0.79** | **0.87** | **43** |
| YAF\_angry | **0.96** | **0.96** | **0.96** | **49** |
| YAF\_disgust | **0.95** | **1.00** | **0.97** | **37** |
| YAF\_fear | **0.92** | **0.89** | **0.90** | **37** |
| YAF\_happy | **1.00** | **0.98** | **0.99** | **52** |
| YAF\_neutral | **0.95** | **0.95** | **0.95** | **39** |
| YAF\_pleasant\_surprised | **0.93** | **0.93** | **0.93** | **40** |
| YAF\_sad | **0.98** | **0.96** | **0.97** | **47** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy** |  |  | **0.93** | **560** |
| Macro avg | **0.93** | **0.93** | **0.93** | **560** |
| Weighted avg | **0.93** | **0.93** | **0.93** | **560** |



**8. Prediction Functionality**

A prediction system can accept .wav files to extract MFCC features which triggers the model to give emotional predictions based on its training. The prediction functionality implements the identical preprocessing and data reshaping operations from training which guarantees data consistency.

**9. Challenges Encountered**

The project encountered several obstacles consisting of different audio file duration ranges as well as maintaining equal feature extraction processes across all samples and managing an unbalanced dataset. Accurate identification becomes more challenging for pleasant surprise and neutral categories since their acoustic signals show overlapping patterns.

**10. Results**

The final model achieves good overall performance, correctly identifying most emotion classes with high accuracy. Some misclassifications occur between similar-sounding emotions like **fear** and **surprise**, but the confusion matrix shows a generally strong classification performance across the dataset.

**11. Future Work**

The system could benefit from upcoming improvements through the utilization of chroma analysis or spectral contrast or from testing hybrid CNN-LSTM structure and leveraging pre-trained audio model features. Applications of pitch shifting along with noise addition through data augmentation offer prospects to enhance generalization performance. The model would become more accessible for real-time and web-based applications when it is deployed either as an emotion detection web service or as a part of a real-time system.

**12. Conclusion**

This project verifies a complete information pipeline that recognizes emotions through speech based on deep learning methods. The system delivers accurate emotional category classification through the combination of MFCC features with LSTM modeling in its architecture. Further system development alongside real-world utilization will establish such systems as fundamental components for developing emotionally intelligent machines that improve user domain interaction.

Bottom of Form