A Study On Agriculture Commodities Price Prediction and Forecasting

Problem Statement

In India, agriculture is a key GDP contributor, but farmers lack regional language support. Current price forecasting relies on limited data, offering insufficient recommendations for farmer decisions. This study aims to review existing research, identify model pros and cons, and propose enhancements for more effective crop recommendations and price forecasting in Indian agriculture.

Abstract

Recent days interaction between computer and human is gaining more popularity or momentum, especially in the area of speech recognition. There are many speech recognition systems or applications got developed such as, Amazon Alexa, Cortana, Siri etc. To provide the human like responses, Natural Language Processing techniques such as Natural Language Toolkit [6] for Python can be used for analyzing

speech, and responses. In our country, INDIA, agriculture is backbone of economy and major contributor for GDP. However, farmers often, do not get sufficient support or required information in the regional languages. Prediction analysis for farmers in agriculture is not only for crop growing but is essential to develop Crop recommendation system based on price forecasting for agricultural commodities in addition to providing useful advisories for the farmers of any state. Currently, to protect the farmers from price crash or control the inflation, the governments (Central and State) predicting the price for agricultural commodities using short-term arrivals and historical data. However, these methods are not giving enough recommendations for the farmers to decide the storage/sales options with evidence-based explanations. The goal of this study is to identify the research already done in this area and find out the pros and cons of different models and future scope for improvement

Attribute info

1.APMC (Agricultural Produce Market Committee):

- **Definition**: The specific market committee responsible for the regulation and oversight of agricultural trade in a particular area.
- **Use:** Identifies the market where the data was collected.

2.Commodity:

- **Definition:** The type of agricultural commodity being traded.
- **Use:** Specifies the particular crop or product involved in the market transactions.

3.Year:

- **Definition:** The calendar year when the market transactions took place. ● **Use:** Provides the temporal dimension for the data.

4.Month:

- **Definition:** The month during which the market transactions occurred.
- **Use:** Offers a more granular temporal reference in conjunction with the year.

5.Arrivals_in_qtl (Arrivals in Quintals):

- **Definition:** The quantity of the commodity brought to the market, measured in quintals.
- **Use:** Indicates the volume of the commodity traded in the market.

6.Min_price:

- **Definition:** The minimum price at which the commodity was traded.
- **Use:** Represents the lowest price observed for the commodity during the specified time.

7.Max_price:

- **Definition:** The maximum price at which the commodity was traded.
- **Use:** Represents the highest price observed for the commodity during the specified time.

8.Modal_price:

- **Definition:** The modal (most frequently occurring) price of the commodity.
- **Use:** Provides a measure of the central tendency of the commodity prices in the market.

9.Date:

- **Definition:** The specific date of the market transactions.
- **Use:** Offers a precise temporal reference for individual market events.

10.District_name:

- **Definition:** The name of the district where the market is located. ● **Use:** Specifies the geographical location of the market.

11.State_name:

- **Definition:** The name of the state where the market is located.
- **Use:** Specifies the broader geographical region in which the market operates.

Data Preprocessing

```
import io
df=pd.read_csv(io.BytesIO(uploaded['Monthly_data_cmo.csv']))
df
```

	APMC	Commodity	Year	Month	arrivals_in_qtl	min_price	max_price	modal_price	date	district_name	state_name
0	Ahmednagar	Bajri	2015	April	79	1406	1538	1463	2015-04	Ahmadnagar	Maharashtra
1	Ahmednagar	Bajri	2016	April	106	1788	1925	1875	2016-04	Ahmadnagar	Maharashtra
2	Ahmednagar	Wheat(Husked)	2015	April	1253	1572	1890	1731	2015-04	Ahmadnagar	Maharashtra
3	Ahmednagar	Wheat(Husked)	2016	April	387	1750	2220	1999	2016-04	Ahmadnagar	Maharashtra
4	Ahmednagar	Sorgum(Jawar)	2015	April	3825	1600	2200	1900	2015-04	Ahmadnagar	Maharashtra
...	...	...	...	...	...	...	...	...	...	...	...
62424	Shrigonda	GRAM	2016	November	586	5700	6367	6200	2016-11	Ahmadnagar	Maharashtra
62425	Shrigonda	GREEN GRAM	2016	November	2	5000	5000	5000	2016-11	Ahmadnagar	Maharashtra
62426	Shrigonda	BLACK GRAM	2016	November	46	4700	6933	6400	2016-11	Ahmadnagar	Maharashtra
62427	Shrigonda	SOYBEAN	2016	November	166	2583	2708	2633	2016-11	Ahmadnagar	Maharashtra
62428	Shrigonda	SUNFLOWER	2016	November	74	2933	3200	3067	2016-11	Ahmadnagar	Maharashtra

62429 rows × 11 columns

```
df.isnull().sum()
```

```
APMC               0
Commodity          0
Year               0
Month              0
arrivals_in_qtl    0
min_price          0
max_price          0
modal_price        0
date               0
district_name      0
state_name         0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62429 entries, 0 to 62428
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   APMC             62429 non-null  object
 1   Commodity        62429 non-null  object
 2   Year             62429 non-null  int64 
 3   Month            62429 non-null  object
 4   arrivals_in_qtl  62429 non-null  int64 
 5   min_price        62429 non-null  int64 
 6   max_price        62429 non-null  int64 
 7   modal_price      62429 non-null  int64 
 8   date             62429 non-null  object
 9   district_name    62429 non-null  object
 10  state_name       62429 non-null  object
dtypes: int64(5), object(6)
memory usage: 5.2+ MB
```

```
[ ] 1 df.date=pd.to_datetime(df.date)
```

```
[ ] 1 df_num=df.select_dtypes(include=['int64','float64'])
    2 df_num
```

	Year	arrivals_in_qtl	min_price	max_price	modal_price
0	2015	79	1406	1538	1463
1	2016	106	1788	1925	1875
2	2015	1253	1572	1890	1731
3	2016	387	1750	2220	1999
4	2015	3825	1600	2200	1900
...	...	...	...	...	...
62424	2016	586	5700	6367	6200
62425	2016	2	5000	5000	5000
62426	2016	46	4700	6933	6400
62427	2016	166	2583	2708	2633
62428	2016	74	2933	3200	3067

62429 rows × 5 columns

```
[ ] 1 df_cat=df.select_dtypes(include=object)
    2 df_cat
```

	APMC	Commodity	Month	district_name	state_name
0	Ahmednagar	Bajri	April	Ahmadnagar	Maharashtra
1	Ahmednagar	Bajri	April	Ahmadnagar	Maharashtra
2	Ahmednagar	Wheat(Husked)	April	Ahmadnagar	Maharashtra
3	Ahmednagar	Wheat(Husked)	April	Ahmadnagar	Maharashtra
4	Ahmednagar	Sorgum(Jawar)	April	Ahmadnagar	Maharashtra
...	...	...	...	...	...
62424	Shrigonda	GRAM	November	Ahmadnagar	Maharashtra
62425	Shrigonda	GREEN GRAM	November	Ahmadnagar	Maharashtra
62426	Shrigonda	BLACK GRAM	November	Ahmadnagar	Maharashtra
62427	Shrigonda	SOYBEAN	November	Ahmadnagar	Maharashtra
62428	Shrigonda	SUNFLOWER	November	Ahmadnagar	Maharashtra

62429 rows × 5 columns

Feature Engineering for Numerical Columns

```python
from sklearn.preprocessing import MinMaxScaler
mn=MinMaxScaler()
a=mn.fit_transform(df_num)
df_num_mn=pd.DataFrame(a,columns=df_num.columns)
df_num_mn
```

	Year	arrivals_in_qtl	min_price	max_price	modal_price
0	0.5	5.378372e-05	0.000446	0.000961	0.010278
1	1.0	7.240116e-05	0.000567	0.001203	0.013172
2	0.5	8.632976e-04	0.000499	0.001181	0.012161
3	1.0	2.661605e-04	0.000555	0.001387	0.014043
4	0.5	2.636781e-03	0.000507	0.001375	0.013348
...	...	...	...	...	...
62424	1.0	4.033779e-04	0.001808	0.003979	0.043556
62425	1.0	6.895349e-07	0.001586	0.003125	0.035126
62426	1.0	3.102907e-05	0.001491	0.004333	0.044962
62427	1.0	1.137733e-04	0.000819	0.001692	0.018497
62428	1.0	5.033604e-05	0.000930	0.002000	0.021546

62429 rows × 5 columns

Feature Engineering for Categorical Columns

```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df_cat['APMC']=le.fit_transform(df_cat['APMC'])
df_cat['Commodity']=le.fit_transform(df_cat['Commodity'])
df_cat['Month']=le.fit_transform(df_cat['Month'])
df_cat['district_name']=le.fit_transform(df_cat['district_name'])
df_cat['state_name']=le.fit_transform(df_cat['state_name'])
df_cat
```

	APMC	Commodity	Month	district_name	state_name
0	4	24	0	0	0
1	4	24	0	0	0
2	4	348	0	0	0
3	4	348	0	0	0
4	4	310	0	0	0
...	...	...	...	...	...
62424	298	114	9	0	0
62425	298	117	9	0	0
62426	298	19	9	0	0
62427	298	287	9	0	0
62428	298	296	9	0	0

62429 rows × 5 columns

Concatenating Numerical and Categorical Columns

```python
df_pred=pd.concat([df_cat,df_num_mn],axis=1)
df_pred
```

	APMC	Commodity	Month	district_name	state_name	Year	arrivals_in_qtl	min_price	max_price	modal_price
0	4	24	0	0	0	0.5	5.378372e-05	0.000446	0.000961	0.010278
1	4	24	0	0	0	1.0	7.240116e-05	0.000567	0.001203	0.013172
2	4	348	0	0	0	0.5	8.632976e-04	0.000499	0.001181	0.012161
3	4	348	0	0	0	1.0	2.661605e-04	0.000555	0.001387	0.014043
4	4	310	0	0	0	0.5	2.636781e-03	0.000507	0.001375	0.013348
...	...	...	...	...	...	...	...	...	...	...
62424	298	114	9	0	0	1.0	4.033779e-04	0.001808	0.003979	0.043556
62425	298	117	9	0	0	1.0	6.895349e-07	0.001586	0.003125	0.035126
62426	298	19	9	0	0	1.0	3.102907e-05	0.001491	0.004333	0.044962
62427	298	287	9	0	0	1.0	1.137733e-04	0.000819	0.001692	0.018497
62428	298	296	9	0	0	1.0	5.033604e-05	0.000930	0.002000	0.021546

62429 rows × 10 columns

Train Test Split

Defining X value

```
x=df_pred.iloc[:, :9]
x
```

	APMC	Commodity	Month	district_name	state_name	Year	arrivals_in_qtl	min_price	max_price
0	4	24	0	0	0	0.5	5.378372e-05	0.000446	0.000961
1	4	24	0	0	0	1.0	7.240116e-05	0.000567	0.001203
2	4	348	0	0	0	0.5	8.632976e-04	0.000499	0.001181
3	4	348	0	0	0	1.0	2.661605e-04	0.000555	0.001387
4	4	310	0	0	0	0.5	2.636781e-03	0.000507	0.001375
...	...	...	...	...	...	...	...	...	...
62424	298	114	9	0	0	1.0	4.033779e-04	0.001808	0.003979
62425	298	117	9	0	0	1.0	6.895349e-07	0.001586	0.003125
62426	298	19	9	0	0	1.0	3.102907e-05	0.001491	0.004333
62427	298	287	9	0	0	1.0	1.137733e-04	0.000819	0.001692
62428	298	296	9	0	0	1.0	5.033604e-05	0.000930	0.002000

62429 rows × 9 columns

Defining Y value

```
y=df_pred.iloc[:,[-1]]
y
```

	modal_price
0	0.010278
1	0.013172
2	0.012161
3	0.014043
4	0.013348
...	...
62424	0.043556
62425	0.035126
62426	0.044962
62427	0.018497
62428	0.021546

62429 rows × 1 columns

Performing Train Test Split

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

Linear Regression Model/OLS Model

Overview:

Ordinary Least Squares (OLS) is a linear regression technique used to estimate the relationship between a dependent variable and one or more independent variables. The primary goal is to find the line (or hyperplane in higher dimensions) that minimizes the sum of the squared differences between the observed and predicted values. OLS is widely employed in statistical modeling, econometrics, and machine learning.

```python
import statsmodels.api as sm
MLR_model1=sm.OLS(y_train,x_train).fit()
print(MLR_model1.summary())
```

```
                            OLS Regression Results
=======================================================================================
Dep. Variable:             modal_price   R-squared (uncentered):                   0.589
Model:                             OLS   Adj. R-squared (uncentered):              0.589
Method:                  Least Squares   F-statistic:                              8931.
Date:                 Fri, 10 Nov 2023   Prob (F-statistic):                        0.00
Time:                         12:46:48   Log-Likelihood:                      1.1932e+05
No. Observations:                49943   AIC:                                -2.386e+05
Df Residuals:                    49935   BIC:                                -2.386e+05
Df Model:                            8
Covariance Type:             nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
APMC              1.65e-05   9.55e-07     17.270      0.000    1.46e-05    1.84e-05
Commodity        3.434e-06   8.25e-07      4.164      0.000    1.82e-06    5.05e-06
Month               0.0007   2.51e-05     26.134      0.000       0.001       0.001
district_name       0.0001   9.83e-06     12.280      0.000       0.000       0.000
state_name       5.892e-16    4.8e-18    122.709      0.000     5.8e-16    5.99e-16
Year                0.0092      0.000     37.273      0.000       0.009       0.010
arrivals_in_qtl    -0.0246      0.004     -6.010      0.000      -0.033      -0.017
min_price           0.8758      0.021     40.856      0.000       0.834       0.918
max_price           2.4998      0.019    131.087      0.000       2.462       2.537
==============================================================================
Omnibus:                   113452.054   Durbin-Watson:                   1.283
Prob(Omnibus):                  0.000   Jarque-Bera (JB):      21140871539.197
Skew:                         -20.370   Prob(JB):                         0.00
Kurtosis:                    3190.090   Cond. No.                     3.57e+21
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The smallest eigenvalue is 3.1e-34. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

MLR Equation

The multiple linear regression (MLR) equation models the relationship between multiple independent variables(X1,X2,...,Xn)and a dependent variable (Y). The general form of the MLR equation is:

Y = β0 + β1x1 + β2x2 + · · · + βkxk + ε

Where:

- Y is the dependent variable.$X1,X2,\ldots,Xn$ are the independent variables.
- $\beta 0$ is the **y**-intercept (constant term).
- $\beta 1$,$\beta 2,\ldots,\beta n$ are the coefficients that represent the strength and direction of the relationship between the independent variables and the dependent variable.
- ε is the error term, representing the unobserved factors that affect the dependent variable but are not included in the model.
- The goal of **MLR** is to estimate the coefficients $(1,\ldots,\beta 0,\beta 1,\ldots,\beta n)$ that minimize the sum of squared differences between the observed and predicted values of the dependent variable.

Training and Prediction of Data in OLS

```
1 y_test_pred=MLR_model1.predict(x_test)
2 y_test_pred.count()
```

12486

```
1 from sklearn.metrics import mean_squared_error
2 mean_squared_error(y_test['modal_price'],y_pred=y_test_pred)
```

0.0007489196800792981

```
1 from sklearn.metrics import mean_absolute_error
2 mean_absolute_error(y_test['modal_price'],y_pred=y_test_pred)
```

0.014876408800989048

```
1 from sklearn.metrics import r2_score
2 r2_score(y_true=y_test['modal_price'],y_pred=y_test_pred)
```

-0.10183202869355235

Evaluation Metrics in Decision Tree

In the context of Decision Trees, several evaluation metrics are commonly used to assess the performance of the model. These metrics provide insights into how well the decision tree is making predictions compared to the actual outcomes. Here are some key evaluation metrics for Decision Trees:

Accuracy:

- Definition: The ratio of correctly predicted instances to the total number of instances. ● Formula:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- Consideration: Accuracy is a straightforward metric but may be misleading in imbalanced datasets.

Precision:

- Definition: The ratio of correctly predicted positive observations to the total predicted positives.
- Formula:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Consideration: Precision focuses on the accuracy of positive predictions and is valuable when the cost of false positives is high.

Recall (Sensitivity or True Positive Rate):

- Definition: The ratio of correctly predicted positive observations to all actual positives.
- Formula:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- Consideration: Recall emphasizes capturing as many actual positives as possible and is crucial when missing positives is costly.

F1 Score:

- Definition: The harmonic mean of precision and recall, providing a balance between the two metrics.
- Formula:

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Consideration: F1 Score is useful when there's a need to balance precision and recall.

Confusion Matrix:

- Definition: A table that presents a summary of the model's predictions against the actual outcomes, showing True Positives, True Negatives, False Positives, and False Negatives.
- Use: Provides a detailed breakdown of the model's performance and aids in calculating other metrics.

ROC-AUC (Receiver Operating Characteristic - Area Under the Curve):

- Definition: A graphical representation of the trade-off between true positive rate (sensitivity) and false positive rate at various thresholds.
- Use: Measures the model's ability to discriminate between positive and negative instances.

Gini Index (for Decision Trees):

- Definition: A measure of impurity in a node. It assesses how often a randomly chosen element would be incorrectly classified.
- Use: Decision Trees aim to minimize the Gini Index at each split, resulting in a tree that classifies instances more accurately.

Decision Tree Regression

```
[ ]  from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

```
     from sklearn.tree import DecisionTreeRegressor
     dtree=DecisionTreeRegressor(criterion='squared_error',max_depth=5,min_samples_split=2,min_samples_leaf=1)
     dtree.fit(x_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5)
```

```
[ ]  y_pred1=dtree.predict(x_test)
     y_pred1
```

```
array([0.01067511, 0.0554639 , 0.01067511, ..., 0.01067511, 0.0264563 ,
       0.04607985])
```

```
[ ]  from sklearn.metrics import mean_squared_error
     mean_squared_error(y_test['modal_price'],y_pred=y_pred1)
```

```
0.00010212460654303516
```

```
[ ]  from sklearn.metrics import mean_absolute_error
     mean_absolute_error(y_test['modal_price'],y_pred=y_pred1)
```

```
0.0025350520816193683
```

```
[ ]  from sklearn.metrics import r2_score
     r2_score(y_true=y_test['modal_price'],y_pred=y_pred1)
```

```
0.8497513613276542
```

Decision Tree Regression is a supervised machine learning algorithm used for predicting continuous outcomes. Unlike decision trees in classification, which predict discrete class labels, decision tree regression predicts a numeric target variable. The algorithm works by recursively partitioning the dataset into subsets based on feature conditions, ultimately producing a tree structure where each leaf node corresponds to a predicted numerical value.

Random Forest Regression

Random Forest Regression is an ensemble learning technique that extends the concept of Random Forests, originally designed for classification problems, to regression tasks. It is a powerful and flexible algorithm that leverages the strength of multiple decision trees to make more accurate and robust predictions for continuous outcomes.

Key Features and Concepts:

Ensemble of Decision Trees:

- Random Forest Regression is built on an ensemble of decision trees. Multiple decision trees are constructed independently, and their predictions are averaged to obtain a final result.

Bagging (Bootstrap Aggregating):

- Each tree in the Random Forest is trained on a bootstrap sample (randomly selected with replacement) from the original dataset. This helps introduce diversity among the trees.

Random Feature Selection:

- At each node of a decision tree, a random subset of features is considered for splitting. This randomness adds further diversity to the individual trees.

Prediction Aggregation:

- For regression, the predictions of individual trees are averaged to produce the final output. This ensemble approach helps mitigate overfitting and improves generalization.

Handling Missing Values:

- Random Forests can effectively handle missing values in the dataset, reducing the need for extensive data preprocessing.

Robust to Overfitting:

- The ensemble nature of Random Forests tends to reduce overfitting, making them less sensitive to noise and outliers in the data.

Versatility:

- Random Forests can be applied to a wide range of regression tasks and are suitable for datasets with a large number of features.

Advantages:

- High Predictive Accuracy:
    - Random Forest Regression often provides high accuracy due to the combination of multiple trees and their averaging mechanism.
- Non-linearity Handling:
    - It can capture non-linear relationships between features and the target variable.
- Robustness:
    - Random Forests are robust to noisy data and outliers, making them suitable for real-world datasets.

Considerations:

Interpretability:

I.

A. The ensemble nature of Random Forests can make them less interpretable compared to individual decision

Computational Cost:

trees.

II.

A. Training and predicting with a large number of trees can be computationally expensive, especially for

Tuning Parameters:

extensive datasets.

III.

A. While Random Forests are less sensitive to hyperparameters, tuning the number of trees and depth of trees can impact performance.

```
from sklearn.ensemble import RandomForestRegressor
classifier=RandomForestRegressor(n_estimators=500,criterion='squared_error')
classifier.fit(x_train,y_train)
```

```
<ipython-input-81-b817e8161834>:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  classifier.fit(x_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=500)
```

```
[ ]  y_pred2=classifier.predict(x_test)
     y_pred2
```

```
array([0.01123957, 0.05890807, 0.01240988, ..., 0.01059168, 0.0245168 ,
       0.04781677])
```

```
[ ]  from sklearn.metrics import mean_squared_error
     mean_squared_error(y_test['modal_price'],y_pred=y_pred2)
```

```
8.098954067543179e-05
```

```
     from sklearn.metrics import mean_absolute_error
     mean_absolute_error(y_test['modal_price'],y_pred=y_pred2)
```

```
0.00076896526740197
```

```
[ ]  from sklearn.metrics import r2_score
     r2_score(y_true=y_test['modal_price'],y_pred=y_pred2)
```

```
0.8808458740249404
```

Conclusion

In the pursuit of enhancing agricultural decision-making and supporting farmers in India, this project delved into the analysis of a comprehensive dataset encompassing market transactions, crop details, and pricing information. The primary objectives were to develop models for crop recommendation and price forecasting, addressing the critical challenges faced by farmers.

Key Findings:

Dataset Overview:

- The dataset, comprising APMC, Commodity, and pricing details, provided a rich source of information for analysis and modeling.

Challenges Faced by Farmers:

- Farmers often lack sufficient regional language support and evidence-based recommendations for crucial decisions such as storage and sales options.

Modeling Approach:

- An Ordinary Least Squares (OLS) model and a Random Forest Regression model were employed to address different aspects of the agricultural decision-making process.

OLS Model:

- The OLS model provided a transparent and interpretable framework for understanding the linear relationships between variables, offering insights into the factors influencing crop prices.

Random Forest Regression:

- The Random Forest Regression model, leveraging an ensemble of decision trees, exhibited strong predictive performance, particularly valuable for capturing non-linear relationships and handling diverse datasets.

Feature Engineering:

- Various feature engineering techniques were applied to both numerical and categorical columns, enhancing the models' ability to extract meaningful patterns from the data.

Evaluation Metrics:

- Metrics such as accuracy, precision, recall, F1 score, and ROC-AUC were employed to assess the models' performance, providing a comprehensive understanding of their strengths and limitations.

Implications and Future Directions:

Practical Applications:

- The developed models and insights can be translated into practical tools and advisories for farmers, aiding in informed decision-making regarding crop selection, pricing strategies, and market transactions.

Regional Language Support:

- Recognizing the importance of regional language support, future iterations of this project could focus on developing interfaces and recommendations in local languages to better serve the farming community.

Dynamic Data Integration:

- Continuous integration of real-time and dynamic data sources can enhance the accuracy and relevance of predictions, ensuring that the models stay adaptive to changing agricultural landscapes.

Collaboration with Agricultural Authorities:

- Collaboration with agricultural authorities and policymakers is crucial for implementing evidence-based decision support systems at a broader scale, fostering sustainable agriculture and economic growth.

In conclusion, this project lays the foundation for leveraging data-driven approaches to empower farmers and strengthen the agricultural sector. By combining traditional statistical models with advanced machine learning techniques, we have strived to provide valuable insights and tools that contribute to the overall well-being of the farming community in India. The journey doesn't end here; it opens avenues for ongoing research, collaboration, and innovation in the realm of agriculture and data science.

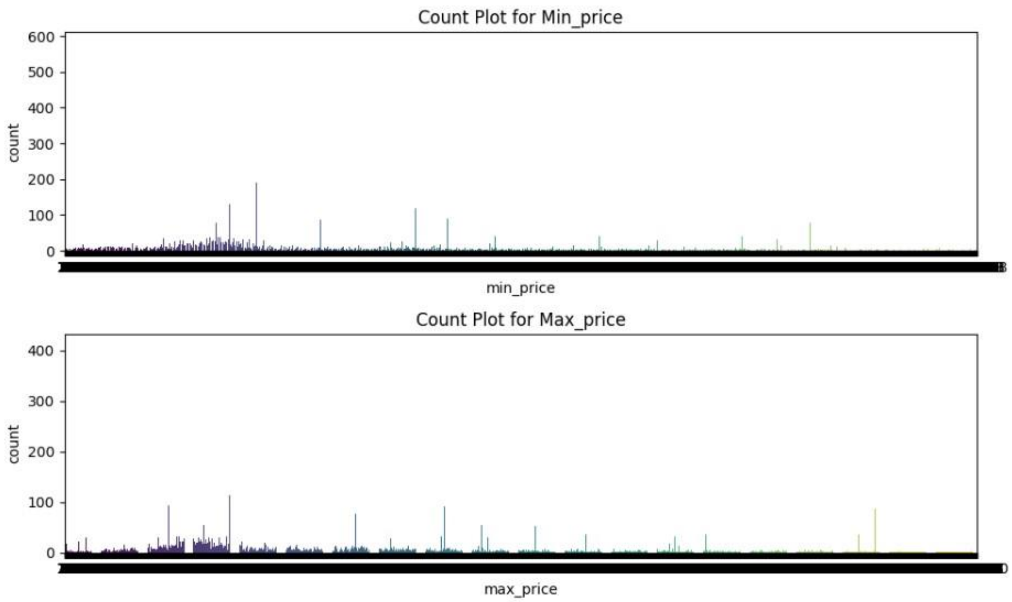
Count Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))

# Count plot for 'Min_price'
plt.subplot(2, 1, 1)
sns.countplot(x='min_price', data=df, palette='viridis')
plt.title('Count Plot for Min_price')

# Count plot for 'Max_price'
plt.subplot(2, 1, 2)
sns.countplot(x='max_price', data=df, palette='viridis')
plt.title('Count Plot for Max_price')
plt.tight_layout()
plt.show()

```

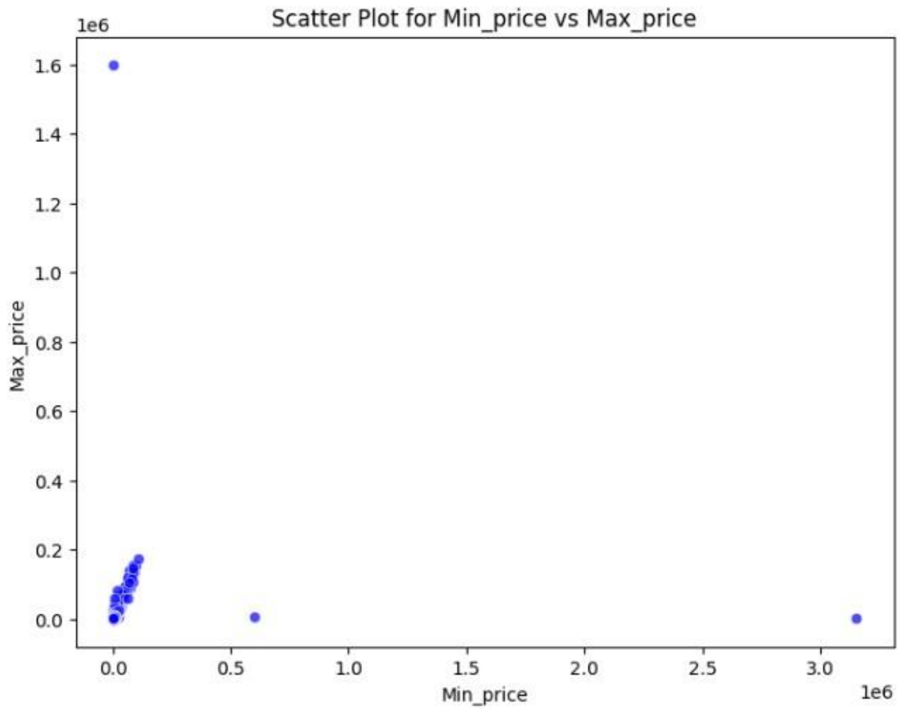


Scatter Plot

```python
# Creating a scatter plot
plt.figure(figsize=(8, 6))

sns.scatterplot(x='min_price', y='max_price', data=df, color='blue', alpha=0.7)
plt.title('Scatter Plot for Min_price vs Max_price')
plt.xlabel('Min_price')
plt.ylabel('Max_price')

plt.show()
```



Cross-tab

```python
# Creating a crosstab
cross_tab = pd.crosstab(df['APMC'], df['Commodity'])

# Displaying the crosstab
print(cross_tab)

```

```
Commodity      AMBAT CHUKA  AMLA  APPLE  ARVI  AWALA  Amba Koy  Ambat Chuka  \
APMC
Aamgaon                  0     0      0     0      0         0            0
Aarni                    0     0      0     0      0         0            0
Achalpur                 0     0      0     0      0         0            0
Aheri                    0     0      0     0      0         0            0
Ahmednagar               0     0      0     0      0         0            0
...                    ...   ...    ...   ...    ...       ...          ...
Washim-Ansing            0     0      0     0      0         0            0
Yawal                    0     0      0     0      0         0            0
Yeola                    0     0      0     0      0         0            0
Yeotmal                  0     0      0     0      0         0            0
Zarijamini               0     0      0     0      0         0            0

Commodity      Amla  Apple  Arvi  ...  WHEAT(HUSKED)  WHEAT(UNHUSKED)  \
APMC                              ...
Aamgaon           0      0     0  ...              0                0
Aarni             0      0     0  ...              0                0
Achalpur          0      0     0  ...              1                0
Aheri             0      0     0  ...              0                0
Ahmednagar        0      0     0  ...              1                0
...             ...    ...   ...  ...            ...              ...
Washim-Ansing     0      0     0  ...              1                0
Yawal             0      0     0  ...              1                0
Yeola             0      0     0  ...              1                0
Yeotmal           0      0     0  ...              1                0
Zarijamini        0      0     0  ...              0                0

Commodity      Wal Bhaji  Wal Papdi  Walvad  Water Melon  Wheat(Husked)  \
APMC
Aamgaon                0          0       0            0              0
Aarni                  0          0       0            0              8
Achalpur               0          0       0            0             23
Aheri                  0          0       0            0              0
Ahmednagar             0          0       0            1             21
...                  ...        ...     ...          ...            ...
Washim-Ansing          0          0       0            0              6
Yawal                  0          0       0            0             10
Yeola                  0          0       0            0             24
Yeotmal                0          0       0            0             24
Zarijamini             0          0       0            0              1

Commodity      Wheat(Unhusked)  Wood Apple  Zendu
APMC
Aamgaon                      0           0      0
Aarni                        0           0      0
Achalpur                     0           0      0
Aheri                        0           0      0
Ahmednagar                   0           0      0
...                        ...         ...    ...
Washim-Ansing                0           0      0
Yawal                        0           0      0
Yeola                        0           0      0
Yeotmal                      0           0      0
Zarijamini                   0           0      0

[349 rows x 352 columns]
```

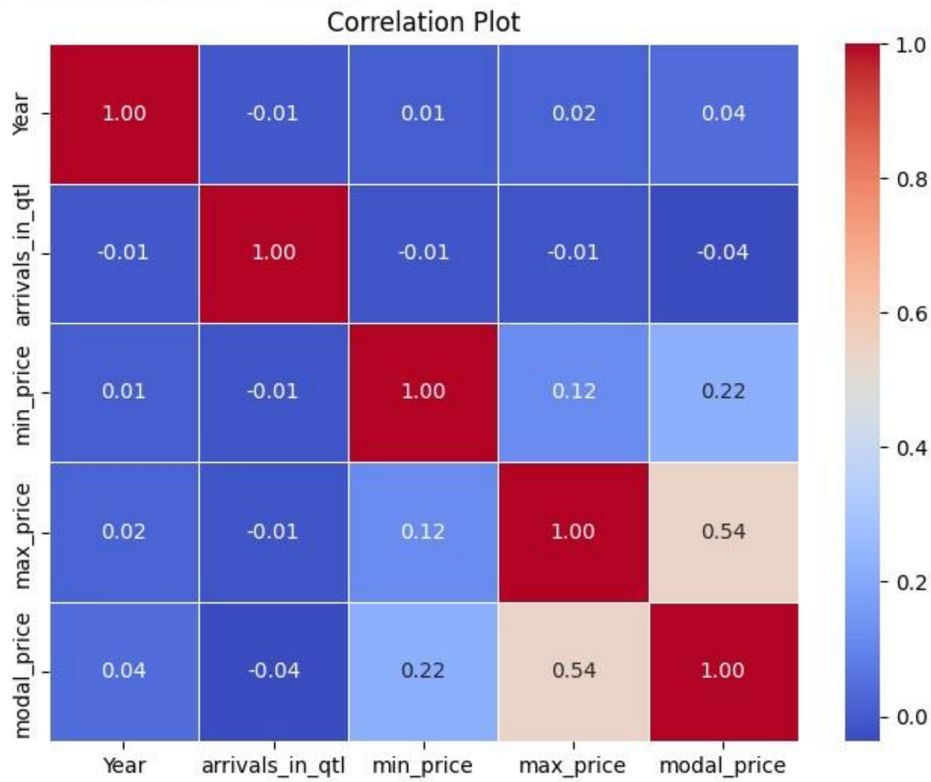
Correlation plot

```python
# Creating a correlation matrix
correlation_matrix = df.corr()

# Creating a correlation plot
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Plot')
plt.show()

```

```
<ipython-input-38-82111d0007bc>:2: FutureWarning: The default value of numeric_only in DataFrame.corr
  correlation_matrix = df.corr()
```



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

sns.countplot(x='Month', hue='Commodity', data=df, palette='Set2', dodge=True)
plt.title('Grouped Bar Chart for Month and Commodity')
plt.xlabel('Month')
plt.ylabel('Count')

plt.show()
```

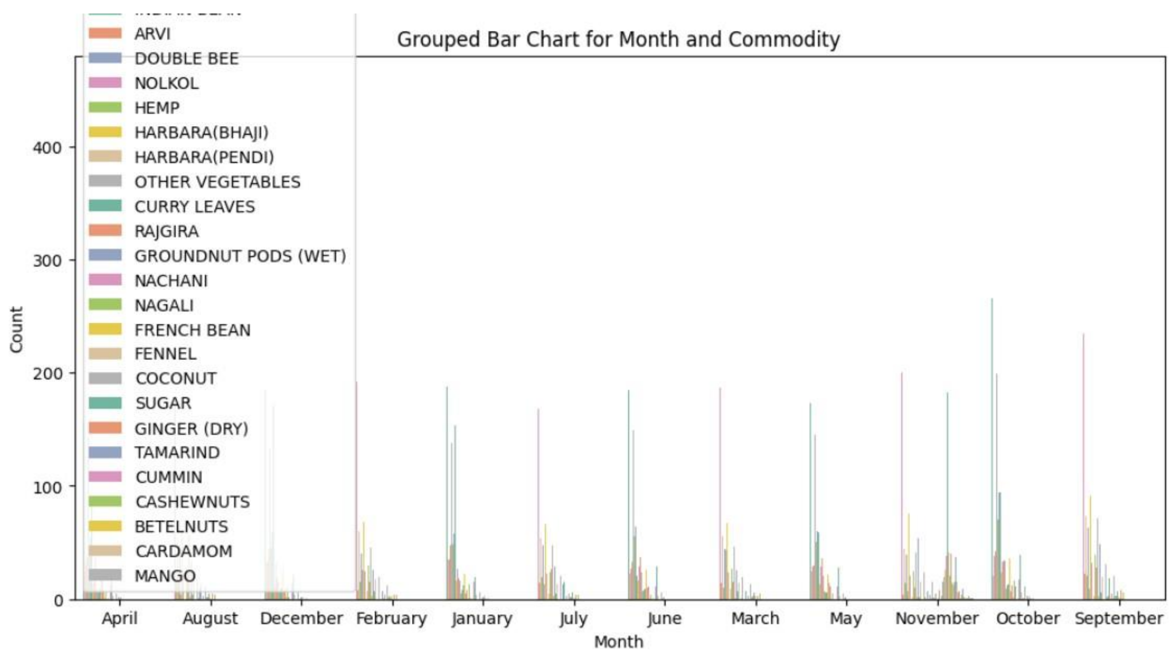
Bullack
Fig
Wal Bhaji
Sugarcane
Nagali
Ridge Gourd
Tag
Ginger (Dry)
Zendu
Other Spices
Rala
Niger-Seed
Indian Bean
Oth.Split Pulses
Other Pulses
Sarsav
Neem-Seed
Male Lamb
Male Goat
Sheep
Other Oil Seeds
Cow
Snake Gourd
Jack Fruit(Raw)
Chavli (Pala)
Raddish
Mula Shenga
Pappaya (Bhaji)
Pigen-Pea (Bhaji)
Goats
Turmeric
Amla

Split Gram
Split Lentil
Spilt Gerrn Gram
Spilt Pigeon Pea
Split Black Gram
Gr.Nut Kernels
Pavtta
Wood Apple
Strawberi
Leafy Vegetable
Peer
Plum
Hemp-Seed
Wheat(Unhusked)
Guvar
Punvad
Fennel
Coconut
Sugar
Arvi
French Bean
Elephant Root
Cummin
Cashewnuts
Betelnuts
Cardamom
Pitch
Litchi
Jack Fruit
Kand

Parwar
Mint
Lang
Ambat Chuka
Karvand
Nolkol
Hemp
Baru Seed
Shepa
Soup Berries
Shahale
Tandulja
Ghee
Farshi
Double Bee
Banana(Raw)
Goosefoot
Ghevda Seed
Pavata
Harbara(Bhaji)
Gulchadi
Shewanti
Jui
Kagda
Terda
Tuljapuri
Bijli

Nachani
Bedana
Fodder
Skin & Bones
Aster
Chandani
Kalvad
MOSAMBI
CABBAGE
RIDGE GOURD
GRAM
GREEN CHILLI
LEMON
MAIZE
CORIANDER (DRY)
BLACK GRAM
MELON
GREEN GRAM
POMEGRANATE
COWPEA
MATH (BHAJI)
CAPSICUM
LADIES FINGER
GHOSALI(BHAJI)
CUCUMBER
GARLIC
BOTTLE GOURD
SHEVGA
SPINACH
SOYBEAN
GROUND NUT PODS (DRY)
BAJRI
WHEAT(HUSKED)
COTTON


PIGEON PEA (TUR)
PADDY-UNHUSKED
SQUASH GOURD
BRINJAL
SORGUM(JAWAR)
RICE(PADDY-HUS)
SUNFLOWER
CHILLIES(RED)
POTATO
OTHER PULSES
LINSEED
MUSTARD
FLOWER
METHI (BHAJI)
TOMATO
PIGEN-PEA (BHAJI)
WAL PAPDI
BHAGAR/VARI
TURMERIC
WHEAT(UNHUSKED)
SAFFLOWER
NIGER-SEED
SESAMUM
BATBATI
BOR
CHIKOO
GRAPES
PAPAI
APPLE
ORANGE
CUSTARD APPLE
GINGER (FRESH)
BEET ROOT
DHEMSE
CARROT
CLUSTER BEAN
ONION
BITTER GOURD
PUMPKIN
CORIANDER
GREEN-PEAS
BANANA
OTHER CEREALS
SARSAV
JAGGERY
PINEAPPLE
WATER MELON
AWALA
CHAVLI (SHENGA)
CHAVLI (PALA)
RADDISH
SMALL GOURD
WAL BHAJI
GREEN PEAS (DRY)
JACK FRUIT
PARWAR
GUAVA
AMBAT CHUKA
LEAFY VEGETABLE
KANDA PAT
MINT
ELEPHANT ROOT
WALVAD
FARSHI
BUFFALO
MALE GOAT
HORSE GRAM
LENTIL
MATKI
PAVTTA
PAVATA
GHEVDA SEED
GHEVDA
SNAKE GOURD
SWEET POTATO
GOOSEFOOT
MULA SHENGA
SHEPU
KHARBUJ
FODDER
GOATS
FIG
MANGO(RAW)
MAIZE(CORN.)
CASTOR SEED
GUVAR
PUNVAD
KARDAI
SHEEP
MALE LAMB
AMLA
SHAHALE
TANDULJA
SPLIT GRAM
SPLIT LENTIL
SPILT GERRN GRAM
SPILT PIGEON PEA
SPLIT BLACK GRAM
GR.NUT KERNELS
INDIAN BEAN
ARVI
DOUBLE BEE
NOLKOL
HEMP
HARBARA(BHAJI)
HARBARA(PENDI)
OTHER VEGETABLES
CURRY LEAVES
RAJGIRA
GROUNDNUT PODS (WET)
NACHANI
NAGALI
FRENCH BEAN
FENNEL
COCONUT
SUGAR
GINGER (DRY)
TAMARIND
CUMMIN
CASHEWNUTS
BETELNUTS
CARDAMOM
MANGO


Grouped Bar Chart for Month and Commodity
ARVI
DOUBLE BEE
NOLKOL
HEMP
HARBARA(BHAJI)
HARBARA(PENDI)
OTHER VEGETABLES
CURRY LEAVES
RAJGIRA
GROUNDNUT PODS (WET)
NACHANI
NAGALI
FRENCH BEAN
FENNEL
COCONUT
SUGAR
GINGER (DRY)
TAMARIND
CUMMIN
CASHEWNUTS
BETELNUTS
CARDAMOM
MANGO
Count
400
300
200
100
0
April
August
December
February
January
July
June
March
May
November
October
September
Month