AVOCADO PRICE PREDICTION

REPORT ON Tech-A-Thon (9th & 10th Nov 2024)

Submitted in fulfillment for the Internal

Assessment of

Artificial Intelligence Theory (BITE308L)

in

B.Tech. – Information Technology

By

Sai Srinivasan M	**(22BIT0036)**
Thrisheiyan U K	**(22BIT0559)**
Sathish K	**(22BIT0672)**
Utkrisht Arya	**(22BIT0190)**
Sudharsanan G	**(22BIT0676)**

Under the guidance of

Dr. S. Hemalatha

SCORE

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science Engineering & Information Systems

Fall Semester 2024-25

TABLE OF CONTENTS

ABSTRACT

This project focuses on predicting avocado prices using machine learning techniques. The dataset, encompassing historical price data based on geographical location, weather changes, and seasonal availability, was analyzed to develop predictive models. Three algorithms—Linear Regression, Random Forest Regressor, and Decision Tree Classifier—were evaluated. Hyperparameter tuning was conducted using RandomizedSearchCV to optimize model performance. Among the models tested, the Random Forest Regressor demonstrated superior predictive accuracy and was selected as the final model.

A user-friendly web application was developed to allow users to input relevant data and receive avocado price predictions in real-time. This solution has potential applications in aiding vendors, producers, and stakeholders in making informed decisions regarding supply chain management and market strategies, ultimately enhancing efficiency and profitability in the avocado industry.

1. Model Description & Details

1.1 Random Forest Regressor

Description:

An ensemble method that builds multiple decision trees during training and outputs their average prediction. It is robust to noise, prevents overfitting, and effectively captures complex relationships between variables.

Tuning and Parameters:

Optimized using **RandomizedSearchCV**, with parameters such as `n_estimators`, `max_depth`, `min_samples_split`, and `max_features` tuned.

Best Parameters Found:

- Number of Estimators: 200
- Max Depth: 30
- Min Samples Split: 2
- Min Samples Leaf: 1
- Max Features: 'log2'

- Bootstrap: False

Performance:

- **R² Score**: 0.89 (highest among the models tested).
- **Mean Absolute Error (MAE)**: 0.10 (lowest error).

1.2 Linear Regression

Description:

A simple, interpretable model assuming a linear relationship between input features and the target variable. It served as the baseline for comparison.

Strengths:

- Easy to implement and understand.
- Efficient on linearly separable data.

Weaknesses:

- Struggles with non-linear relationships in the data, leading to lower accuracy compared to advanced models.

Performance:

- Lower predictive accuracy compared to Random Forest Regressor.

1.3 . Decision Tree Classifier

Description:

- A tree-based algorithm that splits the data into subsets based on feature values. Initially designed for classification tasks, but adapted to regression for this project.

Strengths:

- Simple and interpretable structure.
- Handles non-linear data and captures feature interactions effectively.

Weaknesses:

- Prone to overfitting, especially without pruning or tuning.

Performance:

- Moderate accuracy but not competitive with the Random Forest Regressor due to overfitting and lack of generalization on unseen data.

The Random Forest Regressor emerged as the best model due to its superior accuracy and generalization capabilities.

2. Model design

2.1 Flow Diagram

Data Flow:

2.1.1 **Data Collection**:

Collect avocado sales data, including features such as price, volume, bag size distribution, region, and date.

2.1.2 **Data Preprocessing**:

Convert categorical data (e.g., region) to numeric values using **LabelEncoder**. Extract features like year, month, and day of the week from the Date column. Split data into training and testing sets.

2.1.3 **Model Training & Evaluation**:

Train the following models: **Linear Regression**, **Decision Tree**, and **Random Forest Regressor**. Use **RandomizedSearchCV** for hyperparameter tuning, optimizing model performance based on metrics like

R² Score and **Mean Absolute Error (MAE)**. Select the **best-performing model** (Random Forest Regressor).

2.1.4 **Model Deployment**:

Save the trained model and encoder as .pkl files. Integrate the model into a **web application** for real-time predictions.

2.1.5 **User Interaction**:

User inputs features (e.g., region, date, volume) through the web interface.Predict avocado prices using the deployed model. Display predictions on the web application.

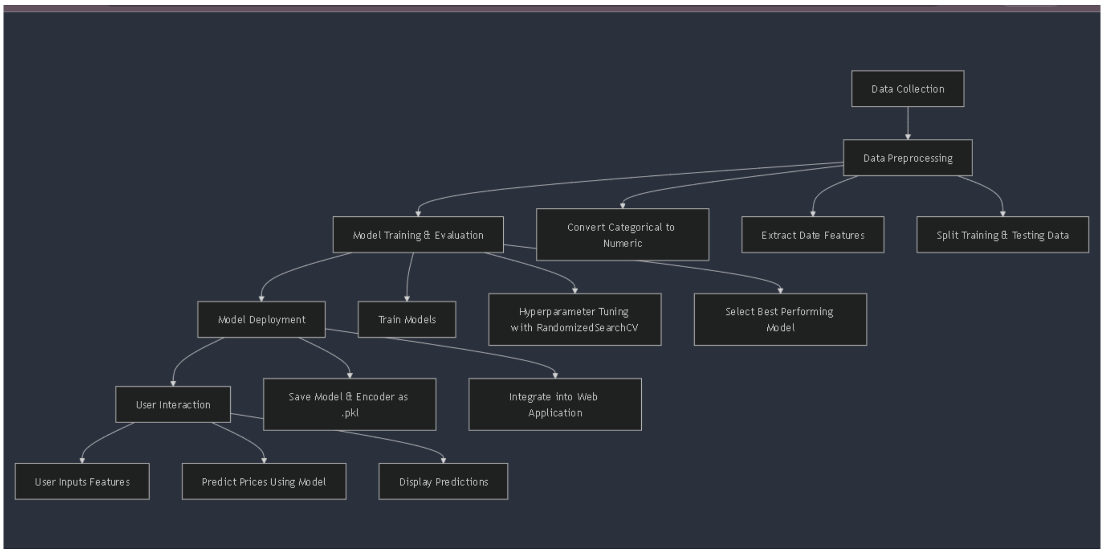

Data Collection
Data Preprocessing
Model Training & Evaluation
Convert Categorical to Numeric
Extract Date Features
Split Training & Testing Data
Model Deployment
Train Models
Hyperparameter Tuning with RandomizedSearchCV
Select Best Performing Model
User Interaction
Save Model & Encoder as .pkl
Integrate into Web Application
User Inputs Features
Predict Prices Using Model
Display Predictions

figure 2.1 flow diagram

3. Software Requirement Specifications

3.1 Introduction

The avocado price prediction system leverages machine learning models to predict avocado prices based on historical data. It provides users with real-time price predictions via a web-based interface.

3.2. Purpose

The system aims to assist producers, vendors, and consumers in forecasting avocado prices, improving decision-making in production and distribution processes.

3.3. Functional Requirements

3.3.1 User Input:

- Users should be able to input relevant details (e.g., region, date, and volume) via a web application.

3.3.2 Data Preprocessing:

- Encode categorical data (e.g., region).
- Extract temporal features (e.g., year, month).

3.3.3 Prediction Model:

- Utilize the trained Random Forest Regressor to predict avocado prices.

3.3.4 Output:

- Display the predicted price on the user interface in real-time.

3.4 Non-Functional Requirements

3.4.1 Performance:

- The system should provide predictions within 2 seconds of user input.

3.4.2 Scalability:

- The system should handle multiple simultaneous requests without significant degradation in performance.

3.4.3 Usability:

- The web interface should be intuitive and user-friendly.

3.4.4 Reliability:

- Ensure model accuracy is consistently high ($R^2 \geq 0.85$).

3.4.5 Security:

- Protect user data and restrict unauthorized access.

3.5 System Requirements

3.5.1 Hardware Requirements:

- **Client-Side**:
    - Minimum: 2 GHz processor, 4 GB RAM, modern web browser (e.g., Chrome, Firefox).
- **Server-Side**:
    - 4 Core CPU, 8 GB RAM, 20 GB storage for model and database.

3.5.2 Software Requirements:

- **Client-Side**:
    - A modern web browser with JavaScript enabled.
- **Server-Side**:
    - Operating System: Windows/Linux/MacOS.
    - Development Frameworks: Flask/Django for the backend, React/Angular for the frontend.
    - Libraries:
        - Python: pandas, numpy, scikit-learn, joblib.
        - Web Server: Nginx/Apache.

3.6 Interfaces

3.6.1 User Interface:

Web application to accept user inputs and display results.

3.7 Constraints

- The system should use only historical avocado sales data available in the dataset.
- Predictions are limited by the accuracy of the Random Forest Regressor and the quality of the data provided.

3.8 Assumptions and Dependencies

- Historical data is clean and updated periodically.
- The Random Forest model remains relevant for predicting future trends.
- Internet connectivity is required for the web application to function.

This SRS provides a clear framework for developing and deploying the avocado price prediction system.

4. Experimental Results & Discussion

The experimental evaluation compared Linear Regression, Decision Tree, and Random Forest Regressor for predicting avocado prices, using metrics like R² Score and Mean Absolute Error (MAE). The Random Forest Regressor outperformed with an R² Score of 0.89 and MAE of 0.10, demonstrating strong accuracy and generalization. Linear Regression struggled with non-linear relationships, and Decision Tree, while better, showed overfitting tendencies. Hyperparameter tuning via RandomizedSearchCV optimized Random Forest's performance, identifying key features like region, total volume, and month as significant predictors. Despite its strengths, the model's effectiveness depends on data quality and may need updates to handle unforeseen market changes. Overall, the Random Forest Regressor proved to be the most reliable model for real-world applications.

4.1 Source code

Data Pre-processing and Model training and comparing :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import  mean_absolute_error,
mean_squared_error,r2_score

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('avocado.csv')

df.drop('Unnamed: 0',axis=1,inplace=True)
df.rename(columns={'4046':'Small_Hass','4225':'Large_Hass','4770':'E
xtralarge_Hass'},inplace=True)

df.to_csv("Mid.csv",index = False)
df = pd.read_csv('Mid.csv')

from scipy.stats import zscore
z =np.abs(zscore(d['AveragePrice']))
print(z)
print(np.where(z<3))
```

```
dn=d[(z<3)]
print('Shape of New Dataframe dn:',dn.shape)

z =np.abs(zscore(dn3['XLarge Bags']))
print(z)
print(np.where(z<3))
dn4=dn3[(z<3)]
print('Shape of New Dataframe dn4:',dn4.shape)

df['Date']=pd.to_datetime(df['Date'])
df['Month']=df['Date'].apply(lambda x:x.month)
df['Day']=df['Date'].apply(lambda x:x.day)

d2=d.copy()
d2.drop(['Date','AveragePrice','Total Volume','region','Total
Bags','Small Bags','Large Bags','XLarge Bags'],axis=1,inplace=True)
d2.groupby(['year']).sum().plot(kind='bar',figsize=(10,5),legend=Tru
e)
plt.title ('Total Hass Avocado Sold')
plt.ylabel('Total Unit Sold')
print('Total Unit Sold
Small_Hass:',(df.groupby(['year'])['Small_Hass'].sum())/1000000)
print('\n')
print('Total Unit Sold
Large_Hass:',(df.groupby(['year'])['Large_Hass'].sum())/1000000)
print('\n')
print('Total Unit Sold
Extralarge_Hass:',(df.groupby(['year'])['Extralarge_Hass'].sum())/10
00000)
plt.show()

d3=d.copy()
d3.drop(['Date','AveragePrice','Total
Volume','Small_Hass','Large_Hass','Extralarge_Hass','region','Total
Bags'],axis=1,inplace=True)
d3.groupby(['year']).sum().plot(kind='bar',figsize=(10,5),legend=Tru
e)
plt.title ('Total Bags Sold (in Million)')
print('Total Small Bags sold (in
Million):',(df.groupby(['year'])['Small Bags'].sum())/1000000)
print('\n')
print('Total Large Bags sold (in
Million):',(df.groupby(['year'])['Large Bags'].sum())/1000000)

x=['conventional','organic']
y=df.groupby('type')['Total Volume'].sum()
plt.figure(figsize=(5,5))
colors=['red','green']
```

```
plt.pie(y,labels=x,shadow=True,colors=colors,explode=(0.3,0),autopct
='%1.1f%%',startangle=45)
plt.axis('equal')
plt.show()

df.drop(columns=["Date"],inplace=True)

le = LabelEncoder()
df['type']= le.fit_transform(df['type'])

df['region'] = pd.Categorical(df['region'])
df_region = pd.get_dummies(df['region'], prefix = 'region')

df = pd.concat([df, df_region], axis=1)
df.drop(columns="region",inplace=True)

df['Month'] = pd.Categorical(df['Month'])
df_month = pd.get_dummies(df['Month'], prefix = 'month')

df = pd.concat([df, df_month], axis=1)
df.drop(columns="Month",inplace=True)

df.to_csv("Final.csv",index = False)

df = pd.read_csv('Final.csv')
X=df.iloc[:,1:78]
y=df['AveragePrice']

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,ran
dom_state=2)
y_test = np.array(y_test,dtype = float)

sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)

def model_accuracy(model,X_train=X_train,y_train=y_train):
    accuracies = cross_val_score(estimator = model, X = X_train, y =
y_train, cv = 5)
    print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
    print("Standard Deviation: {:.2f}
%".format(accuracies.std()*100))

lr=LinearRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
print(y_pred)
print('MAE:', mean_absolute_error(y_test, y_pred))
print('MSE:', mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))

acc_lr=r2_score(y_test,y_pred)
print(acc_lr)

dtr=DecisionTreeRegressor(random_state=3)
dtr.fit(X_train,y_train)
pred=dtr.predict(X_test)

print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))

acc_dtr=r2_score(y_test,pred)
acc_dtr

from sklearn.ensemble import RandomForestRegressor
rdr =
RandomForestRegressor(n_estimators=150,max_features='log2',random_st
ate=42)
rdr.fit(X_train,y_train)
pred1=rdr.predict(X_test)

print('MAE:', mean_absolute_error(y_test, pred1))
print('MSE:', mean_squared_error(y_test, pred1))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred1)))

acc_rdr=r2_score(y_test,pred1)
acc_rdr
```

model.py: random forest regressor

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split,
RandomizedSearchCV
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.preprocessing import LabelEncoder
import joblib

# Load the dataset
data = pd.read_csv("avocado.csv")

# Encode the 'region' column
region_encoder = LabelEncoder()
```

```
data['region'] = region_encoder.fit_transform(data['region'])

# Convert 'Date' to datetime and extract additional features
data['Date'] = pd.to_datetime(data['Date'])
data['year'] = data['Date'].dt.year
data['month'] = data['Date'].dt.month
data['day_of_week'] = data['Date'].dt.dayofweek

# Prepare features and target
X = data[['year', 'month', 'day_of_week', 'Total Volume', 'Total Bags',
'Small Bags', 'Large Bags', 'XLarge Bags', 'region']]
y = data['AveragePrice']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Define the parameter grid for RandomizedSearchCV
param_dist = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [None, 5, 10, 20, 30],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': ['auto', 'sqrt', 'log2'],
    'bootstrap': [True, False]
}

# Initialize the RandomForestRegressor model
rf_model = RandomForestRegressor(random_state=42)

# Perform randomized search with cross-validation
random_search = RandomizedSearchCV(estimator=rf_model,
param_distributions=param_dist,
                                   n_iter=100, cv=3,
scoring='neg_mean_absolute_error',
                                   verbose=1, random_state=42, n_jobs=-
1)

random_search.fit(X_train, y_train)

# Display the best parameters and train the model with them
print("Best parameters found: ", random_search.best_params_)
best_rf_model = random_search.best_estimator_

# Evaluate the model
y_pred = best_rf_model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f"Mean Absolute Error: {mae:.2f}")
print(f"R² Score: {r2:.2f}")

# Save the best model and the encoder
joblib.dump(best_rf_model, 'best_random_forest_model.pkl')
joblib.dump(region_encoder, 'region_encoder.pkl')

print("Model and encoder saved as 'best_random_forest_model.pkl' and
'region_encoder.pkl'")
```

app.py : connect model with ftont end

```
from flask import Flask, render_template, request
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
import joblib
from datetime import datetime

app = Flask(__name__)

model = joblib.load('random_forest_model.pkl')
region_encoder = joblib.load('region_encoder.pkl')

features = ['year', 'month', 'day_of_week', 'Total Volume', 'Total
Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'region']

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        date_str = request.form['date']
        date = datetime.strptime(date_str, '%Y-%m-%d')
        year = date.year
        month = date.month
        day_of_week = date.weekday()

        total_volume = float(request.form['total_volume'])
        total_bags = float(request.form['total_bags'])
        small_bags = float(request.form['small_bags'])
        large_bags = float(request.form['large_bags'])
        xlarge_bags = float(request.form['xlarge_bags'])
        region = request.form['region'].strip()
        region_mapping = {
            'California': 0,
            'New York': 1,
            'Albany': 3,
            'Atlanta': 4,
            'BaltimoreWashington': 5,
            'Boise': 6,
            'Boston': 7,
            'BuffaloRochester': 8,
            'Charlotte': 9,
            'Chicago': 10,
            'CincinnatiDayton': 11,
            'Columbus': 12,
            'DallasFtWorth': 13,
            'Denver': 14,
            'Detroit': 15,
            'GrandRapids': 16,
            'GreatLakes': 17,
            'HarrisburgScranton': 18,
            'HartfordSpringfield': 19,
            'Houston': 20,
            'Indianapolis': 21,
            'Jacksonville': 22,
            'LasVegas': 23,
            'LosAngeles': 24,
            'Louisville': 25,
            'MiamiFtLauderdale': 26,
            'Midsouth': 27,
            'Nashville': 28,
            'NewOrleansMobile': 29,
            'Northeast': 30,
            'NorthernNewEngland': 31,
            'Orlando': 32,
            'Philadelphia': 33,
            'PhoenixTucson': 34,
            'Pittsburgh': 35,
            'Plains': 36,
            'Portland': 37,
            'RaleighGreensboro': 38,
            'RichmondNorfolk': 39,
            'Roanoke': 40,
            'Sacramento': 41,
            'SanDiego': 42,
            'SanFrancisco': 43,
            'Seattle': 44,
            'SouthCarolina': 45,
            'SouthCentral': 46,
            'Southeast': 47,
            'Spokane': 48,
            'StLouis': 49,
            'Syracuse': 50,
            'Tampa': 51,
            'TotalUS': 52,
            'West': 53,
            'WestTexNewMexico': 54
        }

        if region not in region_mapping:
            return render_template('index.html',
prediction_text="Region not recognized.")

        region_encoded = region_mapping[region]

        input_data = np.array([[year, month, day_of_week, total_volume,
total_bags, small_bags, large_bags, xlarge_bags, region_encoded]])

        prediction = model.predict(input_data)

        return render_template('index.html',
prediction_text=f'Predicted Average Price: ${prediction[0]:.2f}')

if __name__ == '__main__':
    app.run(debug=True)
```

index.html : - Front end

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>Avocado Cost Prediction</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boots
trap.min.css"
      rel="stylesheet"
    />
    <style>
      body {
        height: 100vh;
        margin: 0;
        font-family: "Arial", sans-serif;
        background-image:
url("https://cdn.pixabay.com/photo/2019/04/10/11/56/watercolor-4116932_1280.png");
        background-size: cover;
        background-position: center;
        background-repeat: no-repeat;
        display: flex;
        align-items: center;
        justify-content: center;
      }

      form {
        max-width: 600px;
        width: 100%;
        background-color: rgba(255, 255, 255, 0.8);
        padding: 30px;
        border-radius: 8px;
        box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
      }

      h2 {
        color: #008ae5;
        font-weight: bold;
        margin-bottom: 20px;
        text-align: center;
      }

      .section-header {
        color: #008ae5;
        font-size: 1.1rem;
        font-weight: 600;
        margin-bottom: 10px;
        border-bottom: 1px solid #e0e0e0;
        padding-bottom: 5px;
      }

      .form-select,
      .form-control {
        border-radius: 5px;
        border: 1px solid #ced4da;
        padding: 10px;
        font-size: 0.95rem;
        transition: border-color 0.3s;
      }
      .form-select:focus,
      .form-control:focus {
        border-color: #57e2e5;
        box-shadow: none;
```

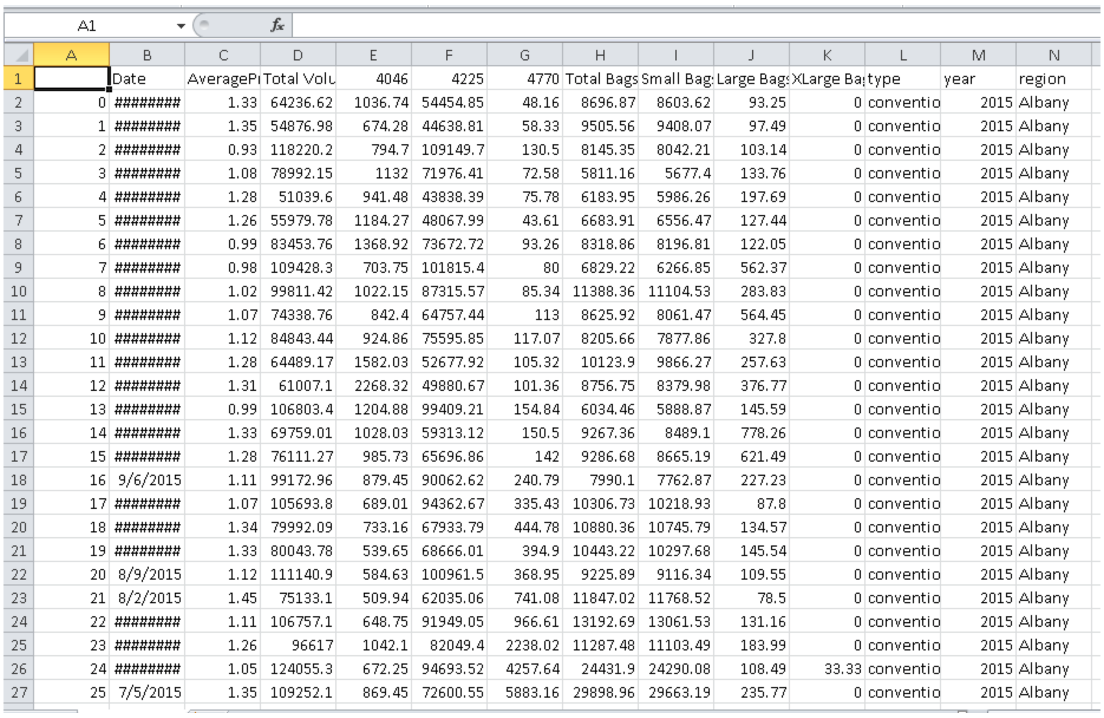
```
    }

    .btn-primary {
      background-color: #008ae5;
      border: none;
      color: #ffffff;
      font-weight: bold;
      transition: background-color 0.3s;
      width: 100%;
      padding: 10px;
    }
    .btn-primary:hover {
      background-color: #46c9c7;
    }

    .note {
      font-size: 0.9rem;
      color: #757575;
      margin-top: 15px;
      text-align: center;
    }
  </style>
</head>
<body>
  <form method="POST" action="/predict">
    <h2>Avocado Cost Prediction</h2>

    <div class="section-header">Date Information</div>
    <div class="mb-3">
      <label for="date" class="form-label">Date</label>
      <input
        type="date"
        class="form-control"
        name="date"
        id="date"
        required
        aria-label="Select date for prediction"
      />
    </div>

    <div class="section-header">Location Information</div>
    <div class="mb-3">
      <label for="region" class="form-label">Region</label>
      <select
        class="form-select"
        name="region"
        id="region"
        required
```

```
          aria-label="Select avocado region"
        >
          <option selected disabled>Select Region</option>
          <option value="California">California</option>
          <option value="New York">New York</option>
          <option value="Albany">Albany</option>
          <option value="Atlanta">Atlanta</option>
          <option
value="BaltimoreWashington">BaltimoreWashington</option>
          <option value="Boise">Boise</option>
          <option value="Boston">Boston</option>
          <option value="BuffaloRochester">BuffaloRochester</option>
          <option value="Charlotte">Charlotte</option>
          <option value="Chicago">Chicago</option>
          <option value="CincinnatiDayton">CincinnatiDayton</option>
          <option value="Columbus">Columbus</option>
          <option value="DallasFtWorth">DallasFtWorth</option>
          <option value="Denver">Denver</option>
          <option value="Detroit">Detroit</option>
          <option value="GrandRapids">GrandRapids</option>
          <option value="GreatLakes">GreatLakes</option>
          <option
value="HarrisburgScranton">HarrisburgScranton</option>
          <option
value="HartfordSpringfield">HartfordSpringfield</option>
          <option value="Houston">Houston</option>
          <option value="Indianapolis">Indianapolis</option>
          <option value="Jacksonville">Jacksonville</option>
          <option value="LasVegas">LasVegas</option>
          <option value="LosAngeles">LosAngeles</option>
          <option value="Louisville">Louisville</option>
          <option value="MiamiFtLauderdale">MiamiFtLauderdale</option>
          <option value="Midsouth">Midsouth</option>
          <option value="Nashville">Nashville</option>
          <option value="NewOrleansMobile">NewOrleansMobile</option>
          <option value="Northeast">Northeast</option>
          <option
value="NorthernNewEngland">NorthernNewEngland</option>
          <option value="Orlando">Orlando</option>
          <option value="Philadelphia">Philadelphia</option>
          <option value="PhoenixTucson">PhoenixTucson</option>
          <option value="Pittsburgh">Pittsburgh</option>
          <option value="Plains">Plains</option>
          <option value="Portland">Portland</option>
          <option value="RaleighGreensboro">RaleighGreensboro</option>
          <option value="RichmondNorfolk">RichmondNorfolk</option>
          <option value="Roanoke">Roanoke</option>
          <option value="Sacramento">Sacramento</option>
```

```html
          <option value="SanDiego">SanDiego</option>
          <option value="SanFrancisco">SanFrancisco</option>
          <option value="Seattle">Seattle</option>
          <option value="SouthCarolina">SouthCarolina</option>
          <option value="SouthCentral">SouthCentral</option>
          <option value="Southeast">Southeast</option>
          <option value="Spokane">Spokane</option>
          <option value="StLouis">StLouis</option>
          <option value="Syracuse">Syracuse</option>
          <option value="Tampa">Tampa</option>
          <option value="TotalUS">TotalUS</option>
          <option value="West">West</option>
          <option value="WestTexNewMexico">WestTexNewMexico</option>
        </select>
      </div>

      <div class="section-header">Product Information</div>
      <div class="mb-3">
        <label for="total_volume" class="form-label">Total
Volume</label>
        <input
          type="number"
          class="form-control"
          name="total_volume"
          id="total_volume"
          placeholder="Enter total volume"
          required
          min="0"
          step="any"
          aria-label="Enter total avocado volume in units"
        />
      </div>
      <div class="mb-3">
        <label for="total_bags" class="form-label">Total Bags</label>
        <input
          type="number"
          class="form-control"
          name="total_bags"
          id="total_bags"
          placeholder="Enter total bags"
          required
          min="0"
          step="any"
          aria-label="Enter total number of bags"
        />
      </div>
      <div class="mb-3">
        <label for="small_bags" class="form-label">Small Bags</label>
        <input
          type="number"
          class="form-control"
          name="small_bags"
          id="small_bags"
          placeholder="Enter small bags"
          required
          min="0"
          step="any"
          aria-label="Enter number of small bags"
        />
      </div>
      <div class="mb-3">
        <label for="large_bags" class="form-label">Large Bags</label>
        <input
          type="number"
          class="form-control"
          name="large_bags"
          id="large_bags"
          placeholder="Enter large bags"
          required
          min="0"
          step="any"
          aria-label="Enter number of large bags"
        />
      </div>
      <div class="mb-3">
        <label for="xlarge_bags" class="form-label">XLarge Bags</label>
        <input
          type="number"
          class="form-control"
          name="xlarge_bags"
          id="xlarge_bags"
          placeholder="Enter XLarge bags"
          required
          min="0"
          step="any"
          aria-label="Enter number of extra-large bags"
        />
      </div>

      <button type="submit" class="btn btn-primary">Predict
Price</button>

      {% if prediction_text %}
      <div class="note">{{ prediction_text }}</div>
      {% else %}
      <div class="note">Enter details to predict the avocado
price</div>
      {% endif %}
    </form>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bun
dle.min.js"></script>
  </body>
</html>
```

4.2.Screenshots with caption

This a raw dataset which is taken from kaggle .

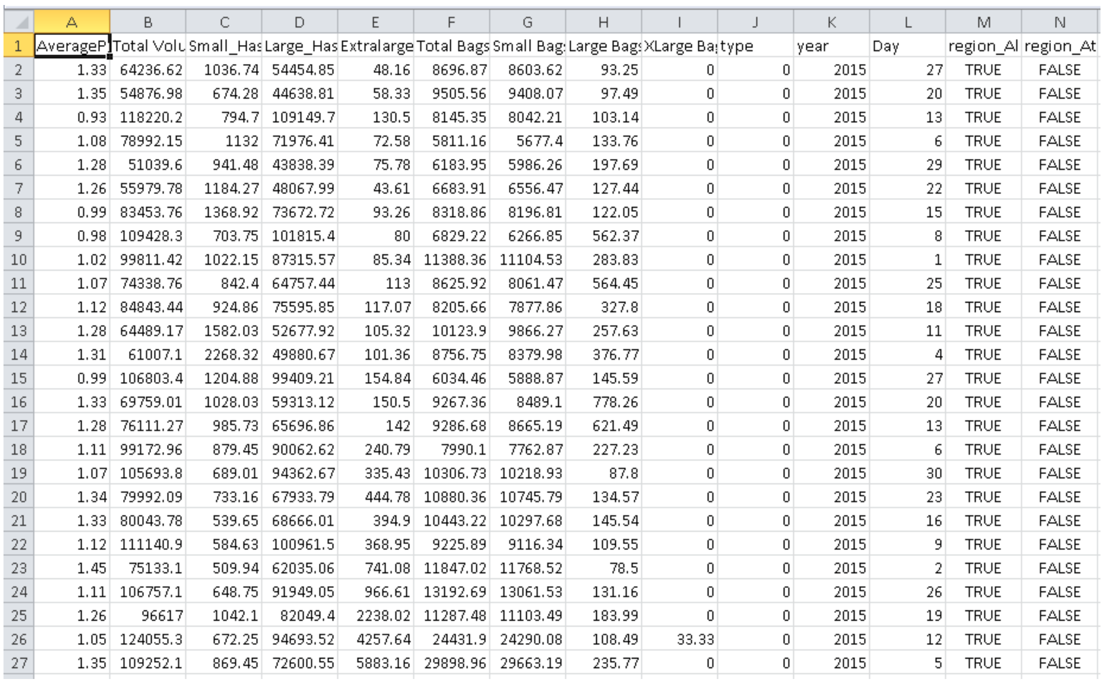


A1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Date	AveragePı	Total Volu	4046	4225	4770	Total Bags	Small Bag:	Large Bag:	XLarge Ba	type	year	region
2	0	########	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0	conventio	2015	Albany
3	1	########	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0	conventio	2015	Albany
4	2	########	0.93	118220.2	794.7	109149.7	130.5	8145.35	8042.21	103.14	0	conventio	2015	Albany
5	3	########	1.08	78992.15	1132	71976.41	72.58	5811.16	5677.4	133.76	0	conventio	2015	Albany
6	4	########	1.28	51039.6	941.48	43838.39	75.78	6183.95	5986.26	197.69	0	conventio	2015	Albany
7	5	########	1.26	55979.78	1184.27	48067.99	43.61	6683.91	6556.47	127.44	0	conventio	2015	Albany
8	6	########	0.99	83453.76	1368.92	73672.72	93.26	8318.86	8196.81	122.05	0	conventio	2015	Albany
9	7	########	0.98	109428.3	703.75	101815.4	80	6829.22	6266.85	562.37	0	conventio	2015	Albany
10	8	########	1.02	99811.42	1022.15	87315.57	85.34	11388.36	11104.53	283.83	0	conventio	2015	Albany
11	9	########	1.07	74338.76	842.4	64757.44	113	8625.92	8061.47	564.45	0	conventio	2015	Albany
12	10	########	1.12	84843.44	924.86	75595.85	117.07	8205.66	7877.86	327.8	0	conventio	2015	Albany
13	11	########	1.28	64489.17	1582.03	52677.92	105.32	10123.9	9866.27	257.63	0	conventio	2015	Albany
14	12	########	1.31	61007.1	2268.32	49880.67	101.36	8756.75	8379.98	376.77	0	conventio	2015	Albany
15	13	########	0.99	106803.4	1204.88	99409.21	154.84	6034.46	5888.87	145.59	0	conventio	2015	Albany
16	14	########	1.33	69759.01	1028.03	59313.12	150.5	9267.36	8489.1	778.26	0	conventio	2015	Albany
17	15	########	1.28	76111.27	985.73	65696.86	142	9286.68	8665.19	621.49	0	conventio	2015	Albany
18	16	9/6/2015	1.11	99172.96	879.45	90062.62	240.79	7990.1	7762.87	227.23	0	conventio	2015	Albany
19	17	########	1.07	105693.8	689.01	94362.67	335.43	10306.73	10218.93	87.8	0	conventio	2015	Albany
20	18	########	1.34	79992.09	733.16	67933.79	444.78	10880.36	10745.79	134.57	0	conventio	2015	Albany
21	19	########	1.33	80043.78	539.65	68666.01	394.9	10443.22	10297.68	145.54	0	conventio	2015	Albany
22	20	8/9/2015	1.12	111140.9	584.63	100961.5	368.95	9225.89	9116.34	109.55	0	conventio	2015	Albany
23	21	8/2/2015	1.45	75133.1	509.94	62035.06	741.08	11847.02	11768.52	78.5	0	conventio	2015	Albany
24	22	########	1.11	106757.1	648.75	91949.05	966.61	13192.69	13061.53	131.16	0	conventio	2015	Albany
25	23	########	1.26	96617	1042.1	82049.4	2238.02	11287.48	11103.49	183.99	0	conventio	2015	Albany
26	24	########	1.05	124055.3	672.25	94693.52	4257.64	24431.9	24290.08	108.49	33.33	conventio	2015	Albany
27	25	7/5/2015	1.35	109252.1	869.45	72600.55	5883.16	29898.96	29663.19	235.77	0	conventio	2015	Albany

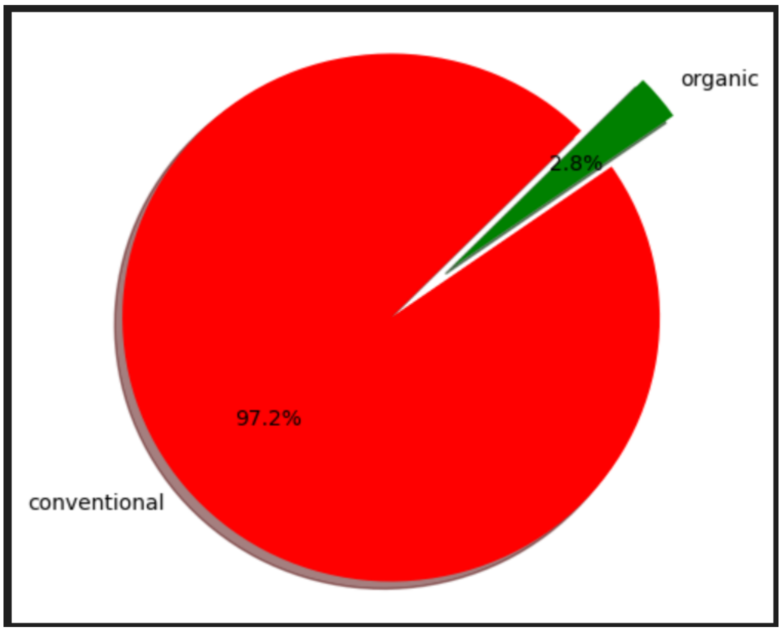
Raw dataset from kaggle

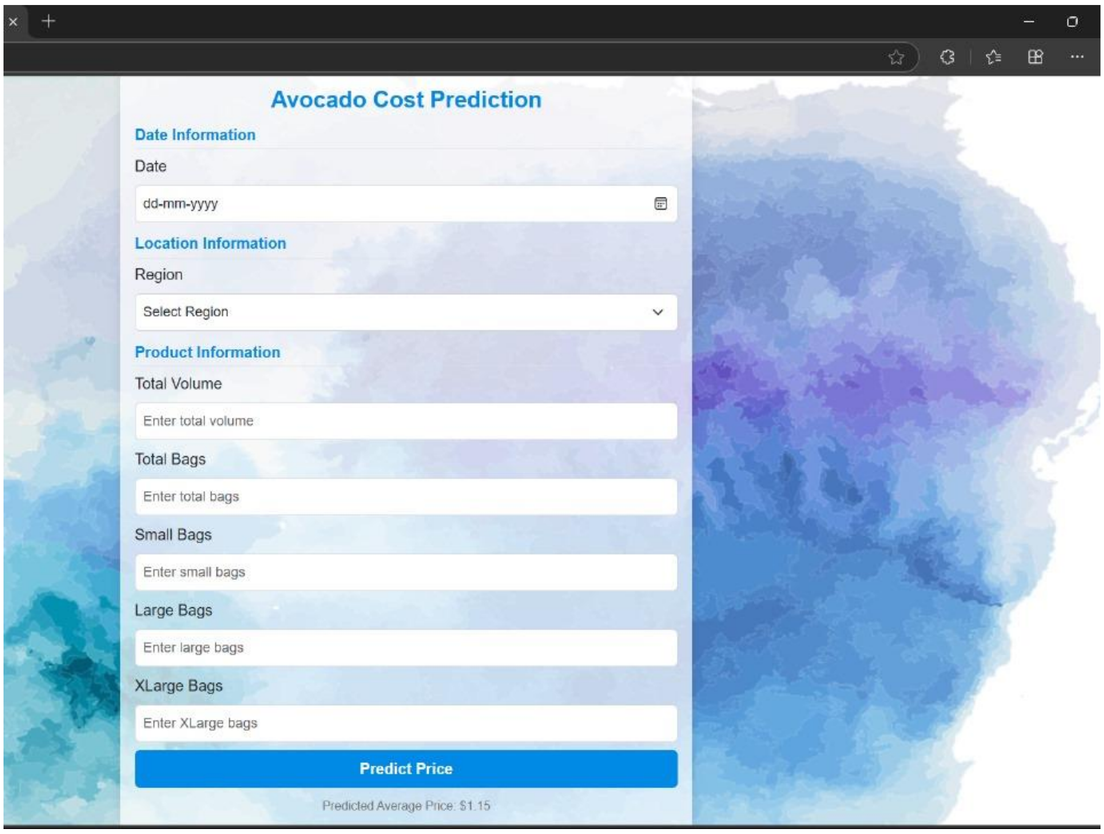
Finalized dataset after the Data-preprocessing .



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	AverageP	Total Volu	Small_Has	Large_Has	Extralarge	Total Bags	Small Bag:	Large Bag:	XLarge Ba	type	year	Day	region_Al	region_At
2	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0	0	2015	27	TRUE	FALSE
3	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0	0	2015	20	TRUE	FALSE
4	0.93	118220.2	794.7	109149.7	130.5	8145.35	8042.21	103.14	0	0	2015	13	TRUE	FALSE
5	1.08	78992.15	1132	71976.41	72.58	5811.16	5677.4	133.76	0	0	2015	6	TRUE	FALSE
6	1.28	51039.6	941.48	43838.39	75.78	6183.95	5986.26	197.69	0	0	2015	29	TRUE	FALSE
7	1.26	55979.78	1184.27	48067.99	43.61	6683.91	6556.47	127.44	0	0	2015	22	TRUE	FALSE
8	0.99	83453.76	1368.92	73672.72	93.26	8318.86	8196.81	122.05	0	0	2015	15	TRUE	FALSE
9	0.98	109428.3	703.75	101815.4	80	6829.22	6266.85	562.37	0	0	2015	8	TRUE	FALSE
10	1.02	99811.42	1022.15	87315.57	85.34	11388.36	11104.53	283.83	0	0	2015	1	TRUE	FALSE
11	1.07	74338.76	842.4	64757.44	113	8625.92	8061.47	564.45	0	0	2015	25	TRUE	FALSE
12	1.12	84843.44	924.86	75595.85	117.07	8205.66	7877.86	327.8	0	0	2015	18	TRUE	FALSE
13	1.28	64489.17	1582.03	52677.92	105.32	10123.9	9866.27	257.63	0	0	2015	11	TRUE	FALSE
14	1.31	61007.1	2268.32	49880.67	101.36	8756.75	8379.98	376.77	0	0	2015	4	TRUE	FALSE
15	0.99	106803.4	1204.88	99409.21	154.84	6034.46	5888.87	145.59	0	0	2015	27	TRUE	FALSE
16	1.33	69759.01	1028.03	59313.12	150.5	9267.36	8489.1	778.26	0	0	2015	20	TRUE	FALSE
17	1.28	76111.27	985.73	65696.86	142	9286.68	8665.19	621.49	0	0	2015	13	TRUE	FALSE
18	1.11	99172.96	879.45	90062.62	240.79	7990.1	7762.87	227.23	0	0	2015	6	TRUE	FALSE
19	1.07	105693.8	689.01	94362.67	335.43	10306.73	10218.93	87.8	0	0	2015	30	TRUE	FALSE
20	1.34	79992.09	733.16	67933.79	444.78	10880.36	10745.79	134.57	0	0	2015	23	TRUE	FALSE
21	1.33	80043.78	539.65	68666.01	394.9	10443.22	10297.68	145.54	0	0	2015	16	TRUE	FALSE
22	1.12	111140.9	584.63	100961.5	368.95	9225.89	9116.34	109.55	0	0	2015	9	TRUE	FALSE
23	1.45	75133.1	509.94	62035.06	741.08	11847.02	11768.52	78.5	0	0	2015	2	TRUE	FALSE
24	1.11	106757.1	648.75	91949.05	966.61	13192.69	13061.53	131.16	0	0	2015	26	TRUE	FALSE
25	1.26	96617	1042.1	82049.4	2238.02	11287.48	11103.49	183.99	0	0	2015	19	TRUE	FALSE
26	1.05	124055.3	672.25	94693.52	4257.64	24431.9	24290.08	108.49	33.33	0	2015	12	TRUE	FALSE
27	1.35	109252.1	869.45	72600.55	5883.16	29898.96	29663.19	235.77	0	0	2015	5	TRUE	FALSE

Finalized dataset





Avocado Cost Prediction

Date Information

Date: dd-mm-yyyy

Location Information

Region: Select Region

Product Information

Total Volume: Enter total volume

Total Bags: Enter total bags

Small Bags: Enter small bags

Large Bags: Enter large bags

XLarge Bags: Enter XLarge bags

Predict Price

Predicted Average Price: $1.15

Web application