**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**



**School of Computer Science Engineering &**

**Information Systems**

Fall Semester 2024-25

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | 3 |
| **1** | **Model Description & Details** | 3 |
| **2** | **Model design** | 5 |
|  | **2.1 Flow Diagram / Flowchart** | 5-6 |
| **3** | **Software Requirement Specifications** | 6-8 |
| **4** | **Experimental Results & Discussion** | 8-9 |
|  | **4.1.Source code** | 9-22 |
|  | **4.2.Screenshots with caption** | 22-24 |

**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**

#### 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.

1. **Model design**

#### ****2.1 Flow Diagram****

#### ****Data Flow:****

* + 1. **Data Collection**:

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

* + 1. **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.

* + 1. **Model Training & Evaluation**:

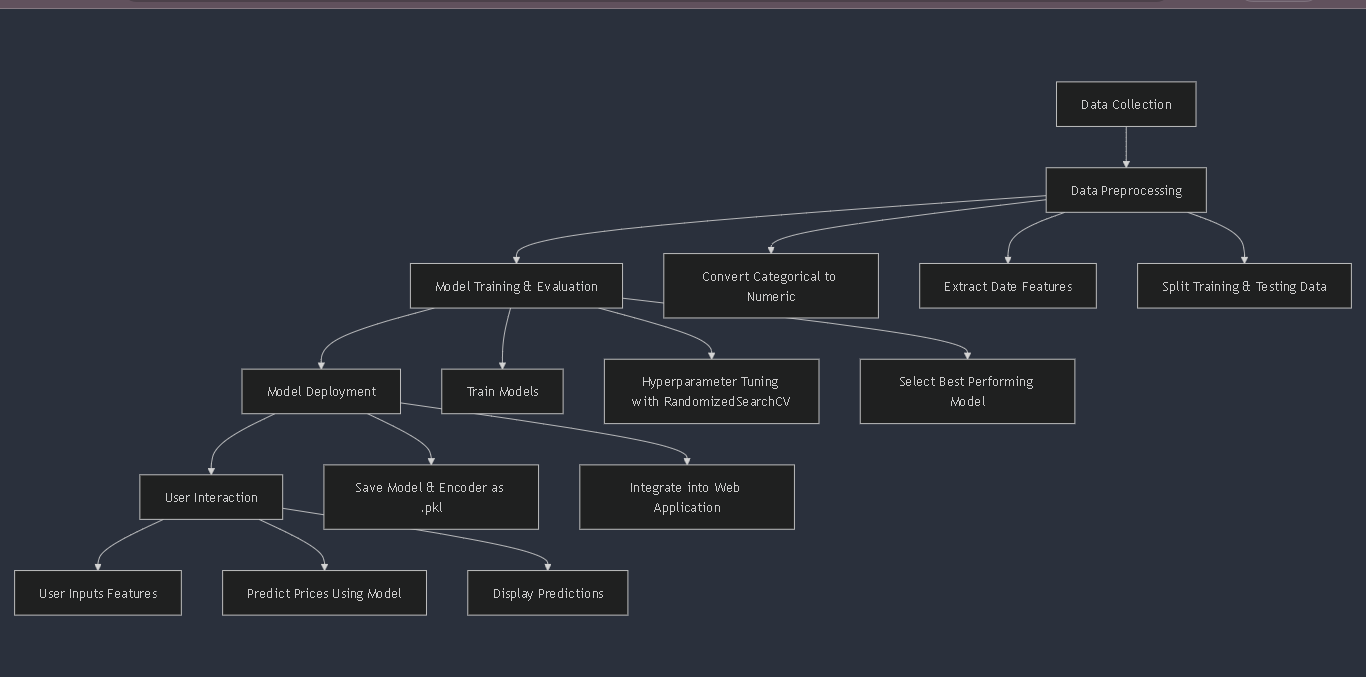
Train the following models: **Linear Regression**, **Decision Tree**, and **Random Forest Regressor**. Use **RandomizedSearchCV** for hyperparameter tuning, optimizing model performance based on metrics like **R² Score** and **Mean Absolute Error (MAE)**. Select the **best-performing model** (Random Forest Regressor).

* + 1. **Model Deployment**:

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

* + 1. **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.

*******figure 2.1 flow diagram*

1. **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****

* + 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.
    1. **Output**:
  + Display the predicted price on the user interface in real-time.

#### ****Non-Functional Requirements****

* + 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 (R² ≥ 0.85).
    1. **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.

1. **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.

* 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':'Extralarge\_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=True)

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())/1000000)

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=True)

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,random\_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\_state=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**

<!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/bootstrap.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>

          <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 %}

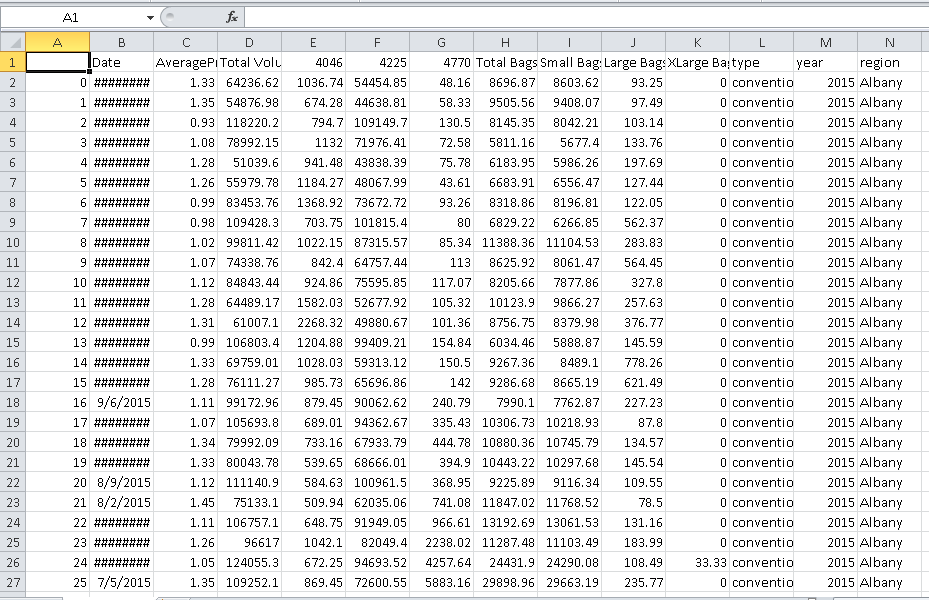
    </form>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

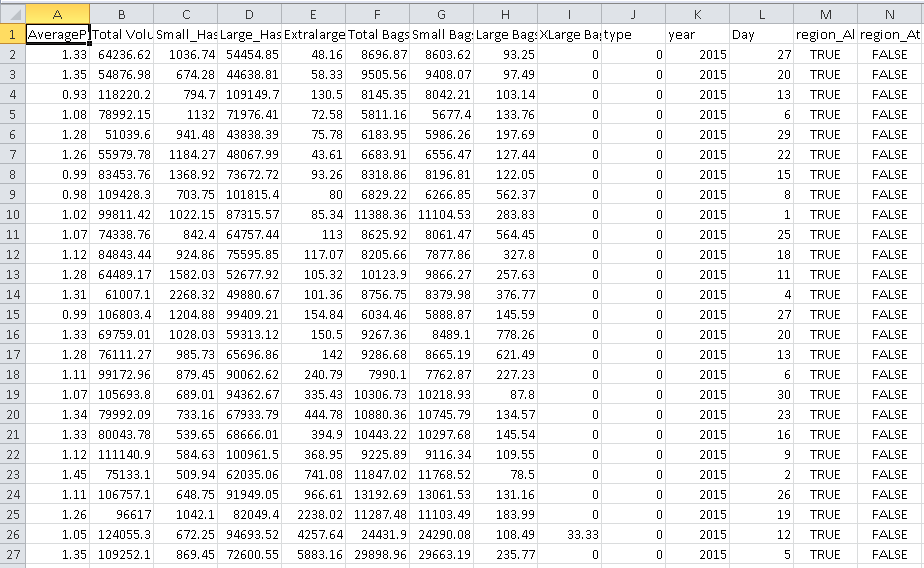
  </body>

</html>

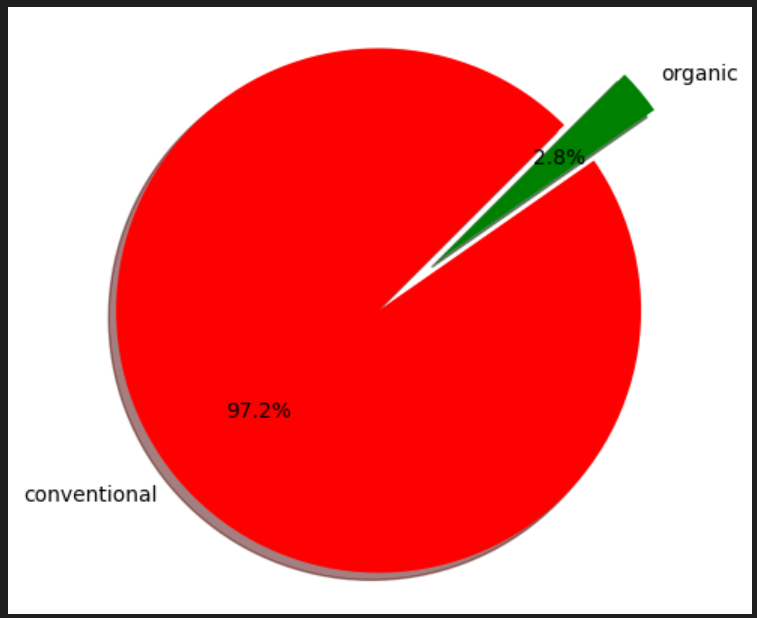
**4.2.Screenshots with caption**

**** This a raw dataset which is taken from kaggle .

*Raw dataset from kaggle*

**** Finalized dataset after the Data-preprocessing .

*Finalized dataset*



*A screenshot of a computer

Description automatically generatedWeb application*