Capstone_E_commerce_Price_Prediction

May 2, 2024

1 ST 1 Assignment 9 Capstone Programming Project (8995)

1.1 E-commerce Price Prediction

Github Link : https://github.com/eliasedwin7/EcommercePricePrediction.git

1.2 Project Team:

- Edwin Alias (u3237781)
- Megha Mary (u3238990)

1.3 Tutorial Group : - 8995 C/06- Thursday 11:30-13:30

1.4 Introduction and Dataset Overview

E-commerce platforms are increasingly reliant on advanced predictive models to optimize product pricing strategies. Accurate price prediction can not only enhance competitiveness but also streamline inventory management and customer satisfaction. This project aims to predict the selling price of e-commerce products using various machine learning models.

1.4.1 1.Setup and Reading the Dataset

[35]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import glob
import tkinter as tk
from tkinter import messagebox
import joblib
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.preprocessing import LabelEncoder
#list all CSV files in the directory
files = glob.glob('data/*.csv')
print(files)
```

```
['data/Test.csv', 'data/Train.csv']
```

[36]:
```python
# Load the dataset
data_train = pd.read_csv(files[1])
print(data_train.head())
```

```
  Product Product_Brand             Item_Category     Subcategory_1  \
0  P-2610         B-659        bags wallets belts              bags
1  P-2453        B-3078                  clothing  women s clothing
2  P-6802        B-1810  home decor festive needs        showpieces
3  P-4452        B-3078  beauty and personal care          eye care
4  P-8454        B-3078                  clothing    men s clothing

       Subcategory_2  Item_Rating        Date  Selling_Price
0          hand bags          4.3    2/3/2017          291.0
1        western wear          3.1    7/1/2015          897.0
2             ethnic          3.5   1/12/2019          792.0
3  h2o plus eye care          4.0  12/12/2014          837.0
4           t shirts          4.3  12/12/2013          470.0
```

[37]:
```python
print(data_train.describe())
```

```
       Item_Rating  Selling_Price
count  2452.000000    2452.000000
mean      3.078467    2494.375612
std       1.187137    7115.256516
min       1.000000      33.000000
25%       2.000000     371.000000
50%       3.100000     596.000000
75%       4.100000    1195.250000
max       5.000000  116289.000000
```

The dataset provided contains details of e-commerce products along with their selling prices. Our analysis will focus on features like `Product_Brand`, `Item_Category`, `Subcategory_1`, `Subcategory_2`, and `Item_Rating`, which are deemed critical for predicting the `Selling_Price`.

1.4.2 2. Problem Statement Definition

Predict the price of ecommerce products based on features such as category, brand, user ratings, etc. This could help sellers on the platform price their products competitively and understand factors influencing prices.

1.4.3 3. Target Variable Identification

[38]:
```python
print(data_train.columns)
```

```
Index(['Product', 'Product_Brand', 'Item_Category', 'Subcategory_1',
       'Subcategory_2', 'Item_Rating', 'Date', 'Selling_Price'],
      dtype='object')
```

[39]:
```python
# Identify the target variable
target_variable = 'Selling_Price'
# Display summary statistics of the target var
print(data_train[target_variable].describe())
```

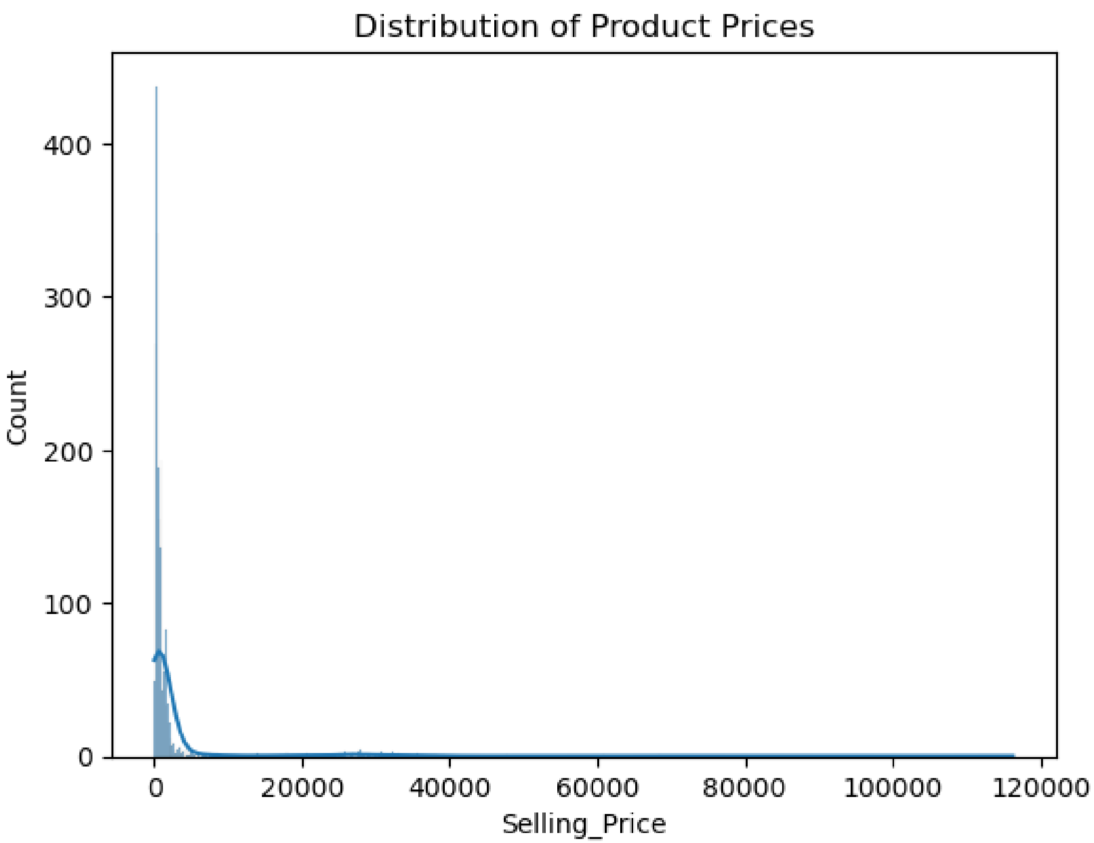
```
count      2452.000000
mean       2494.375612
std        7115.256516
min          33.000000
25%         371.000000
50%         596.000000
75%        1195.250000
max      116289.000000
Name: Selling_Price, dtype: float64
```

1.5 Exploratory Data Analysis (EDA)

Our EDA revealed that the `Selling_Price` is right-skewed, indicating a prevalence of more affordably priced products with fewer high-end outliers. The correlations between subcategories and item ratings with the selling price suggest that these features could be significant predictors in our price prediction model.

1.5.1 4. Visualizing the Distribution of the Target Variable

[40]:
```python
# Visualize the distribution of the target variable
sns.histplot(data_train[target_variable], kde=True)
plt.title('Distribution of Product Prices')
plt.show()
```



1.5.2 5. Basic Level Data Exploration

[41]:
```
# Basic exploration of data
print("Descriptive statistics\n",data_train.describe())
```

```
Descriptive statistics
        Item_Rating  Selling_Price
count  2452.000000    2452.000000
mean      3.078467    2494.375612
std       1.187137    7115.256516
min       1.000000      33.000000
25%       2.000000     371.000000
50%       3.100000     596.000000
75%       4.100000    1195.250000
max       5.000000  116289.000000
```

1.5.3 6. Identifying and Rejecting Unwanted Columns

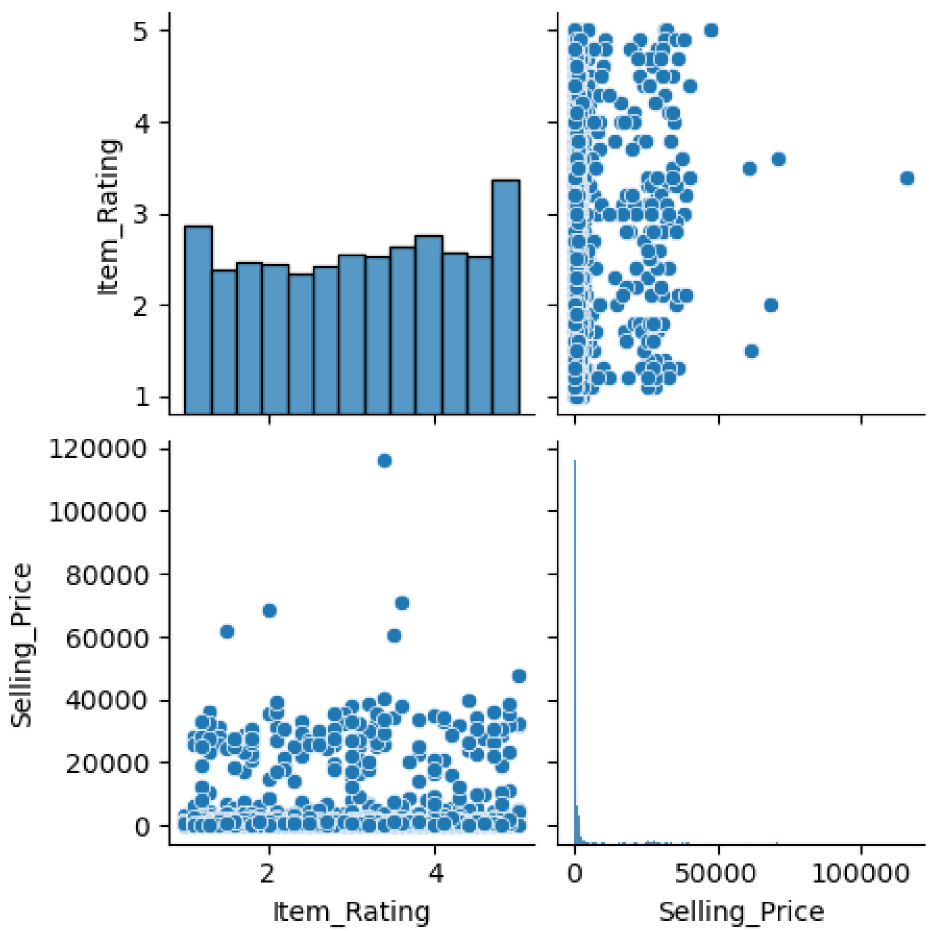
[42]:
```
# Identify columns to drop
columns_to_drop = ['Product', 'Date']
data_train= data_train.drop(columns=columns_to_drop)
print(data_train.columns)
```

```
Index(['Product_Brand', 'Item_Category', 'Subcategory_1', 'Subcategory_2',
       'Item_Rating', 'Selling_Price'],
      dtype='object')
```

1.5.4 7. Visual Exploratory Data Analysis

[43]:
```
# Visual exploratory analysis with pairplot
sns.pairplot(data_train)
plt.show()
```

```
/home/edwin/anaconda3/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



1.5.5 8. Feature Selection Based on Data Distribution

[44]:
```
# Analyze features and select all
selected_features = data_train.columns
print("Selected Features:", selected_features)
```

```
Selected Features: Index(['Product_Brand', 'Item_Category', 'Subcategory_1',
'Subcategory_2',
       'Item_Rating', 'Selling_Price'],
      dtype='object')
```

1.5.6 9. Removal of Outliers and Missing Values

[45]:
```
# Remove outliers and handle missing values
data_train = data_train[data_train[target_variable] <
 ↪data_train[target_variable].quantile(0.99)]
# Forward fill for handling missing values
data_train.fillna(method='ffill', inplace=True)
```

1.5.7 11. Data Conversion to Numeric Values

[46]:
```
# Identify categorical columns
categorical_cols = data_train.select_dtypes(include=['object']).columns
print("Categorical columns:", categorical_cols)
# Initialize the LabelEncoder
label_encoder = LabelEncoder()
# Apply Label Encoding to each categorical column
for column in categorical_cols:
    data_train[column] = label_encoder.fit_transform(data_train[column])
print(data_train.head())
```

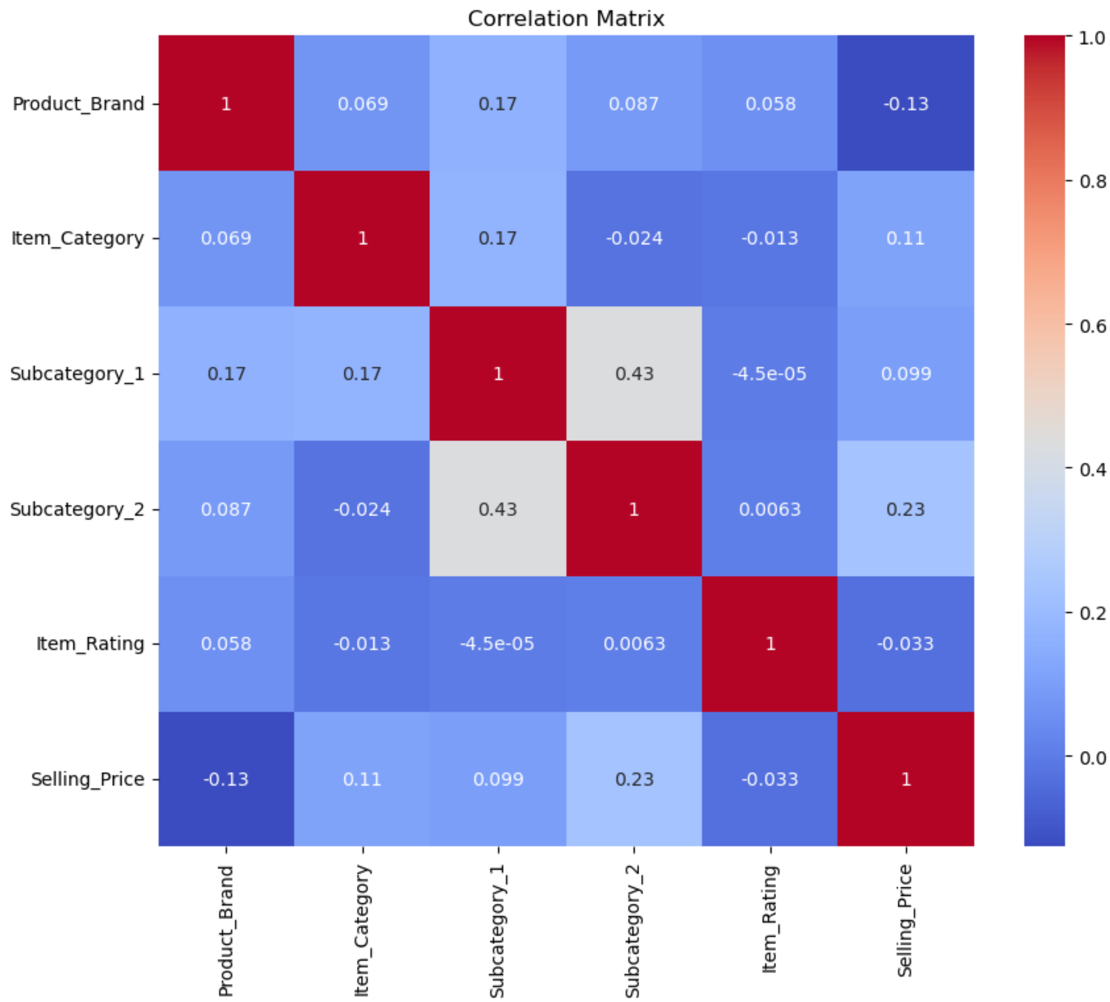
```
Categorical columns: Index(['Product_Brand', 'Item_Category', 'Subcategory_1',
'Subcategory_2'], dtype='object')
   Product_Brand  Item_Category  Subcategory_1  Subcategory_2  Item_Rating  \
0            859              7             10            137          4.3
1            667             10            127            329          3.1
2            280             29            112            101          3.5
3            667              8             37            134          4.0
4            667             10             80            296          4.3

   Selling_Price
0          291.0
1          897.0
2          792.0
```

```
3            837.0
4            470.0
```

1.5.8 10. Visual and Statistical Correlation Analysis

[47]:
```python
# Correlation analysis
plt.figure(figsize=(10, 8))
sns.heatmap(data_train.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



1.6 Predictive Data Analysis

We evaluated three regression algorithms: Linear Regression, Random Forest, and SVM. The Random Forest Regressor emerged as the best model, demonstrating a balance between bias and variance, as reflected in its R^2 score.

1.6.1 12. Training/Testing Sampling and K-Fold Cross Validation

[48]:
```
# Split the data into train and test sets
X = data_train.drop(target_variable, axis=1)
y = data_train[target_variable]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
# Setup K-Fold cross-validation
kf = KFold(n_splits=5)
```

1.6.2 13. Investigating Multiple Regression Algorithms

[49]:
```
# Initialize models
model_lr = LinearRegression()
model_rf = RandomForestRegressor(n_estimators=100, random_state=42)
model_svm = SVR(kernel='linear')

# Function to perform training and cross-validation
def train_and_evaluate(model, X_train, y_train, kf):
    # Perform cross-validation
    scores = cross_val_score(model, X_train, y_train, cv=kf, scoring='r2')
    return np.mean(scores), np.std(scores)

# Evaluate models using K-fold cross-validation
mean_r2_lr, std_r2_lr = train_and_evaluate(model_lr, X_train, y_train, kf)
mean_r2_rf, std_r2_rf = train_and_evaluate(model_rf, X_train, y_train, kf)
mean_r2_svm, std_r2_svm = train_and_evaluate(model_svm, X_train, y_train, kf)

print(f"Linear Regression - Mean R²: {mean_r2_lr:.3f}, Std R²: {std_r2_lr:.3f}")
print(f"Random Forest - Mean R²: {mean_r2_rf:.3f}, Std R²: {std_r2_rf:.3f}")
print(f"SVM - Mean R²: {mean_r2_svm:.3f}, Std R²: {std_r2_svm:.3f}")
```

```
Linear Regression - Mean R²: 0.082, Std R²: 0.031
Random Forest - Mean R²: 0.741, Std R²: 0.054
SVM - Mean R²: -0.069, Std R²: 0.011
```

1.6.3 14. Selection of the Best Model

[16]:
```
model_performance = {
    'Linear Regression': {'Mean R2': mean_r2_lr, 'Std R2': std_r2_lr,'model':
↪model_lr},
    'Random Forest': {'Mean R2': mean_r2_rf, 'Std R2': std_r2_rf,'model':
↪model_rf},
    'SVM': {'Mean R2': mean_r2_svm, 'Std R2': std_r2_svm,'model':model_svm}
}

# Determine the best model based on Mean R² scores
```

```
best_model_name = max(model_performance, key=lambda k:␣
↪model_performance[k]['Mean R2'])
best_model_stats = model_performance[best_model_name]
best_model=best_model_stats['model']

print(f"The best model is {best_model_name}")
```

```
The best model is Random Forest
```

[17]:
```
# Model Training and Validation
# Fitting the model to the Training set
model = best_model
model.fit(X_train, y_train)
# Predicting the Test set results
y_pred = model.predict(X_test)
# Calculating metrics
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```
Mean Absolute Error (MAE): 840.53
Mean Squared Error (MSE): 4489565.65
Root Mean Squared Error (RMSE): 2118.86
R-squared (R2): 0.83
```

These metrics indicate that the model has a strong predictive capability with an R^2 score suggesting that approximately 83% of the variance in the `Selling_Price` is explained by the model.

[50]:
```
# some example based on the training dataset
feature_names = ['Product_Brand', 'Item_Category', 'Subcategory_1',␣
↪'Subcategory_2', 'Item_Rating']
features = [[859, 7, 10, 137, 4.3]]
new_data = pd.DataFrame(features, columns=feature_names)
# Use the best model to make a prediction
predicted_price = best_model.predict(new_data)[0]
print(f"The predicted selling price is: ${predicted_price:.2f}")
```

```
The predicted selling price is: $705.86
```

1.6.4 15. Deployment of the Best Model in Production

The best-performing model was deployed into a simple application using Tkinter, enabling users to input product features and receive price predictions. The GUI is designed to be intuitive, providing a seamless user experience.

[51]:
```
model_name=best_model_name.replace(' ','_')
# Save the model
joblib.dump(best_model, f'{model_name}.pkl')
```

[51]: ['Random_Forest.pkl']

Tkinter Application

[52]:
```
# Load the trained model
model = joblib.load(f'{model_name}.pkl')
# Function to predict the selling price
def predict_price():
    try:
        # Extract values from GUI, convert to floats
        feature_values = [
            float(entry_product_brand.get()),
            float(entry_item_category.get()),
            float(entry_subcategory1.get()),
            float(entry_subcategory2.get()),
            float(entry_item_rating.get())
        ]
        # Create DataFrame with the same feature names as the training set
        feature_names = ['Product_Brand', 'Item_Category', 'Subcategory_1',␣
↪'Subcategory_2', 'Item_Rating']
        input_data = pd.DataFrame([feature_values], columns=feature_names)
        # Predict using the model
        prediction = model.predict(input_data)[0]
        # Display the predicted price
        messagebox.showinfo("Prediction", f"Predicted Selling Price:␣
↪${prediction:.2f}")
    except ValueError:
        # Error handling for invalid inputs
        messagebox.showerror("Input Error", "Please enter valid numbers for all␣
↪input fields")
# the main window
root = tk.Tk()
root.title("E-commerce Price Prediction")
default_values = {
    'Product_Brand': 859,
    'Item_Category': 7,
    'Subcategory_1': 10,
    'Subcategory_2': 137,
    'Item_Rating': 4.3
}
# Function to create labeled entry with default value
def create_labeled_entry(label_text, default_value):
```

```
    frame = tk.Frame(root)
    label = tk.Label(frame, text=label_text)
    label.pack(side=tk.LEFT)
    entry = tk.Entry(frame)
    entry.insert(0, str(default_value))  # Set default value
    entry.pack(side=tk.RIGHT)
    frame.pack(pady=2)
    return entry
# entries for each feature
entry_product_brand = create_labeled_entry('Product Brand',␣
 ↪default_values['Product_Brand'])
entry_item_category = create_labeled_entry('Item Category',␣
 ↪default_values['Item_Category'])
entry_subcategory1 = create_labeled_entry('Subcategory 1',␣
 ↪default_values['Subcategory_1'])
entry_subcategory2 = create_labeled_entry('Subcategory 2',␣
 ↪default_values['Subcategory_2'])
entry_item_rating = create_labeled_entry('Item Rating',␣
 ↪default_values['Item_Rating'])
# Button to predict price
predict_button = tk.Button(root, text="Predict Price", command=predict_price)
predict_button.pack(pady=10)
# Start the application
root.mainloop()
```

1.7 References

1. Bhuwanesh H., "E-commerce Price Prediction," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/bhuwanesh340/ecommerce-price-prediction.
2. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," 2011. [Online]. Available: https://scikit-learn.org/stable/.
3. "pandas: powerful Python data analysis toolkit," pandas development team, 2020. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/index.html.
4. "NumPy," NumPy developers, 2021. [Online]. Available: https://numpy.org/.
5. "Matplotlib: Visualization with Python," Matplotlib developers, 2021. [Online]. Available: https://matplotlib.org/.
6. M. Waskom et al., "seaborn: statistical data visualization," 2020. [Online]. Available: https://seaborn.pydata.org/.