Data Analyst Internship Report

Introduction

Name	Vedant Maladkar
Project Title	• E-commerce Furniture Dataset 2024 • Iris classification
Domain	Data Analyst
Tools	Python, Machine Learning

Project 1- E-commerce Furniture Dataset 2024

Objective-

The aim is to build a logistic regression model that helps us predict whether a furniture product will be a high-selling item (sold > 50 units) on the bases of its price and shipping option.

This model enables e-commerce businesses to identify high-selling products for targeted marketing, better inventory decisions, and competitive pricing strategies.

Dataset Description-

This dataset is named 'E-commerce Furniture Dataset 2024' and it contains historical record of furniture products sold. This dataset contains 2000 entries and multiple columns such as 'productTitle', 'originalPrice', 'price', 'sold', 'tagText'.

Exploratory Data Analysis (EDA)-

```
df[df.isnull().any(axis=1)]
```

✓ 0.0s

	productTitle	originalPrice	price	sold	tagText
0	Dresser For Bedroom With 9 Fabric Drawers Ward...	NaN	$46.79	600	Free shipping
1	Outdoor Conversation Set 4 Pieces Patio Furnit...	NaN	$169.72	0	Free shipping
3	Modern Accent Boucle Chair,Upholstered Tufted ...	NaN	$111.99	0	Free shipping
6	5-Piece Patio Furniture Set Outdoor Couch with...	NaN	$198.31	1	Free shipping
8	TV Stand Dresser For Bedroom With 5 Fabric Dra...	NaN	$43.96	3	NaN
...	...	...	...	...	...
1995	Modern TV Stand Entertainment Center with Two ...	NaN	$72.49	8	Free shipping
1996	Large Wardrobe Armoire Closet with 3 Doors, Fr...	NaN	$276.65	2	Free shipping
1997	Velvet Futon Sofa Bed, 73-inch Sleeper Couch w...	NaN	$196.44	10	Free shipping
1998	Furniture Acrylic Coffee Table Transparent Liv...	NaN	$228.18	0	Free shipping
1999	Bed Frane Bamboo and Metal Platform Bed Frame ...	NaN	$99.48	0	Free shipping

1513 rows × 5 columns

```
df.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   productTitle   2000 non-null   object
 1   originalPrice  487 non-null    object
 2   price          2000 non-null   object
 3   sold           2000 non-null   int64 
 4   tagText        1997 non-null   object
dtypes: int64(1), object(4)
memory usage: 78.3+ KB
```

Using this I remove the 'originalPrice' column as more than 1000 values are null and removed the missing values in 'tagText' column.

```
df = df.dropna(subset=['tagText'])
✓ 0.0s
```

After doing this I got

```
df[df.isnull().any(axis=1)]
✓ 0.0s
```

productTitle	price	sold	tagText	Sales	Shipping

This shows that there are no null values in this dataset.

Feature Engineering-

```
df['Sales'] = df['sold'].apply(lambda x: 1 if x > 50 else 0)
```
✓ 0.0s

```
df['Shipping'] = df['tagText'].apply(lambda x: 1 if 'Free' in str(x) else 0)
```
✓ 0.0s

- This creates a new column 'Sales' where the value in binary form such that (1 if Quantity > 50) or else 0.
- The same goes for 'Shipping' where 'Free Shipping' is 1 and the other is 0.

```
scalar=StandardScaler()
df[['price', 'Shipping']]=scalar.fit_transform(df[['price', 'Shipping']])
```
✓ 0.0s

- This normalizes the 'price' and 'Shipping' using Standard Scalar

```
X = df[['price', 'Shipping']]
X = sm.add_constant(x)

vif_data = pd.DataFrame()
vif_data['feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

print(vif_data)
```
✓ 0.0s

```
    feature        VIF
0     const  17.123073
1     price   1.027422
2  Shipping   1.027422
```

- This uses VIF analysis to remove the multi-colinear features.

Model Building-

We used Logistic Regression to classify products are high-selling or not. The dataset is split in 80-20 training and testing sets. Here classweight is balanced because it automatically adjusts for class imbalance by assigning higher weights to minor class.

```
x_train, x_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=42)
```
✓ 0.0s

```
model = LogisticRegression(class_weight='balanced')
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```
✓ 0.0s

Model Evaluation includes:

- Accuracy Score
- Confusion Matrix
- Classification Matrix

Results-

```
accuracy_score(y_test, y_pred)
```
✓ 0.0s

```
0.725
```

```
cm=confusion_matrix(y_test, y_pred)
cm
```
✓ 0.0s

```
array([[269, 109],
       [  1,  21]])
```

```
print(classification_report(y_test, y_pred, target_names=['Not High', 'High']))
```
✓ 0.0s

```
              precision    recall  f1-score   support

    Not High       1.00      0.71      0.83       378
        High       0.16      0.95      0.28        22

    accuracy                           0.72       400
   macro avg       0.58      0.83      0.55       400
weighted avg       0.95      0.72      0.80       400
```

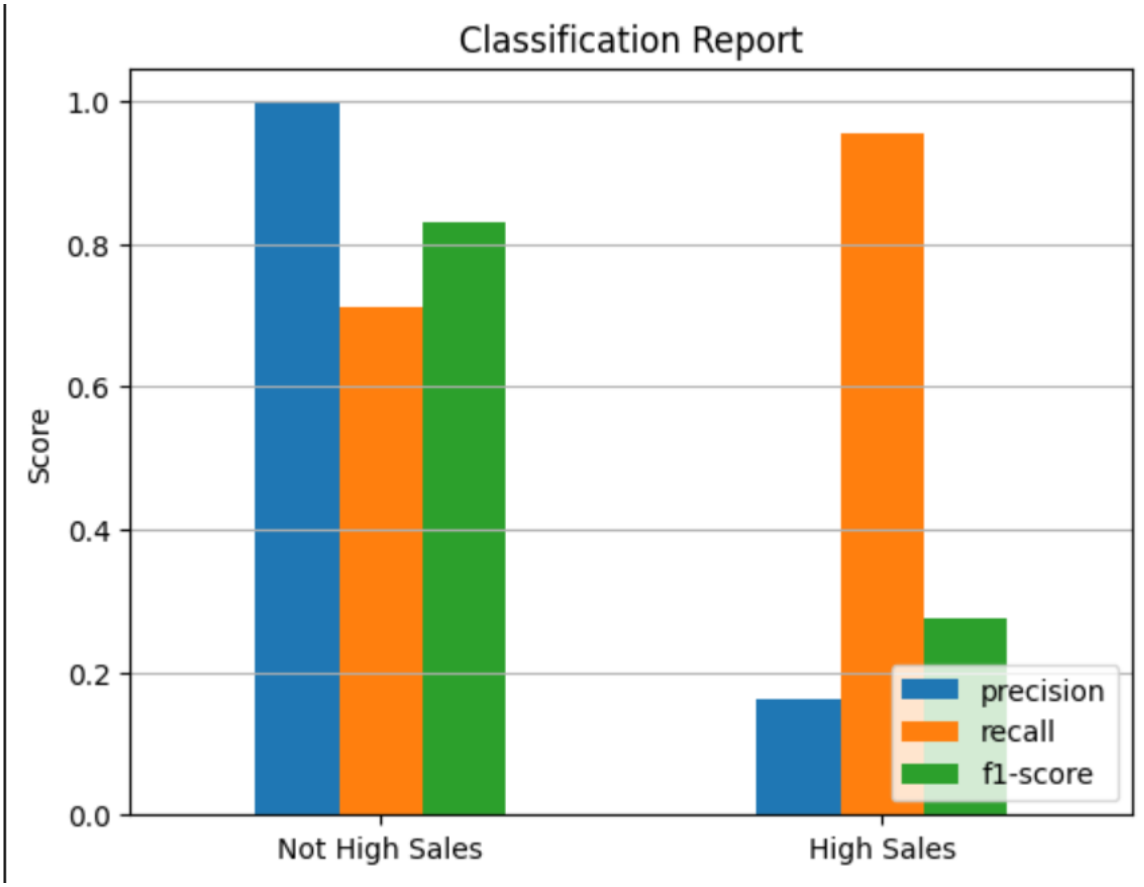
The Logistic Regression shows an accuracy score of approximately 73%

Confusion Matrix and Classification Reports showed:

- Very high recall rate for 'High Sales' which is '0.95'. This model identified 21 out of 22 high-selling products.
- Low precision for 'High Sales' of '0.16'. This tells us that the model flagged many products as high-selling which was wrong, indicating there is room for improving precision by reducing false positives.

Plot-

This plot show the classification report for both not high-selling and high-selling.



Summary-

The model effectively supports decision-making by ensuring high-selling products are correctly flagged, even if it occasional has some false positives. Future improvements can be made by including additional features such as product category, customer review rate, etc.

Reference-

- Libraries - pandas, matplotlib, scikit-learn, statsmodels.
- Tools – Visual Studio Code

Project 2- Iris classification

Objective-

The aim of this project involves creating a Logistic Regression model to classify iris flowers into three species (Setosa, Versicolour, and Virginica) based on the length and width of their petals and sepals.

Problem Statement-

- The model should achieve a high level of accuracy in classifying iris species.
- The model's predictions should be consistent and reliable, as measured by cross-validation.

Dataset Description-

This dataset is named ‘Iris’. It contains a total of 150 entries. The columns of this dataset are 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Species'

Exploratory Data Analysis (EDA)-

```
df.info()
✓ 0.1s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Seeing this we can say that the there are no null values and the dataset is already cleaned.

Feature Engineering-

```
label_encoder = LabelEncoder()
df['Species'] = label_encoder.fit_transform(df['Species'])
```
✓ 0.0s

Using this the code automatically converts the 'Species' string values to the numerical values (iris-setosa, iris-virsicolour, iris-viginica) as 0, 1, 2

```
scalar=StandardScaler()
df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]=scalar.fit_transform(df[['SepalLengthCm',
 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']])
```
✓ 0.0s Python

This normalized 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm' using Standard Scalar.

```
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
X = sm.add_constant(x)

vif_data = pd.DataFrame()
vif_data['feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

print(vif_data)
```
✓ 0.0s

```
         feature        VIF
0          const   1.000000
1  SepalLengthCm   7.103113
2   SepalWidthCm   2.099039
3  PetalLengthCm  31.397292
4   PetalWidthCm  16.141564
```

Here we are removing 'PetalLengthCm' since the value is multicollinear. After removing the column we get

```
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalWidthCm']]
X = sm.add_constant(X)

vif_data = pd.DataFrame()
vif_data['feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

[148] ✓ 0.0s

```
        feature       VIF
0         const  1.000000
1  SepalLengthCm  3.414225
2   SepalWidthCm  1.294507
3   PetalWidthCm  3.864678
```

Model Building-

Here we use Logistic Regression and divide dataset to 80-20 for training and testing sets.

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

✓ 0.0s

Generate + Code + Markdown

```
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

✓ 0.0s

Model evaluation includes:

- Accuracy Score
- Confusion matrix
- R2-Score
- Mean Squared Error
- Classification report
- K-Fold Cross Validation

Results-

```
accuracy_score(y_test, y_pred)
```
✓ 0.0s

```
0.9666666666666667
```

```
cm=confusion_matrix(y_test, y_pred)
cm
```
✓ 0.0s

```
array([[10,  0,  0],
       [ 0,  8,  1],
       [ 0,  0, 11]])
```

Generate

```
r2_score(y_test,y_pred)
```
✓ 0.0s

```
0.9523052464228935
```

```
mse = mean_squared_error(y_test, y_pred)
print(mse)
```
✓ 0.0s

```
0.03333333333333333
```

```
cr=classification_report(y_test,y_pred, target_names=['Setosa', 'versicolor', 'virginica'])
print(cr)
```
✓ 0.0s

```
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.89      0.94         9
   virginica       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```

```
model = LogisticRegression()
K=5
kfold = KFold(K,random_state=0,shuffle=True)
mse_cv = cross_val_score(model,x,y,cv=kfold,scoring='accuracy')
print(mse_cv)
```

✓ 0.1s

```
[1.         0.9        1.         1.         0.93333333]
```

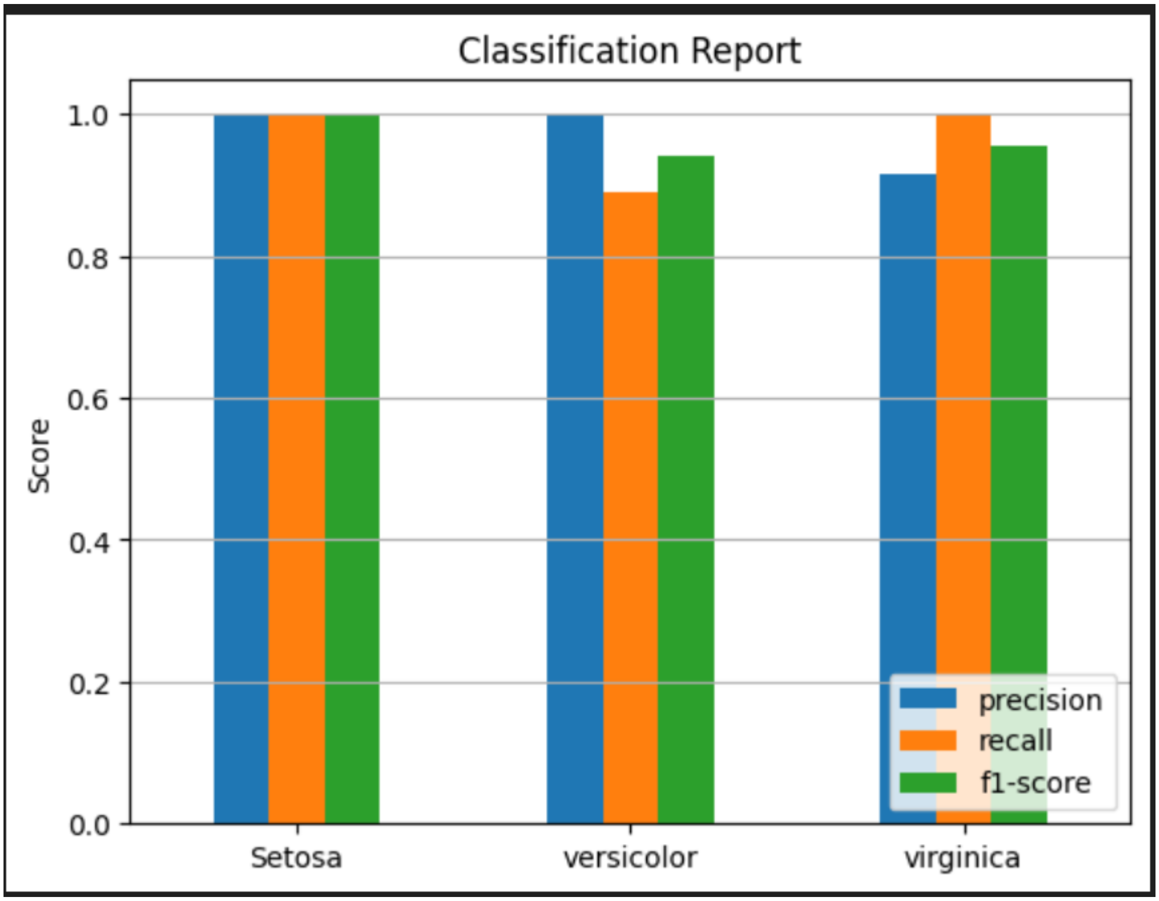
Logistic model accuracy is approximately 97%

Model evaluation shows:

- R2 score is approximately 0.95 showing strong model fit
- Mean Squared Error is 0.033 indicating low prediction error
- Setosa and Virginica showed no false negatives or false positives which means it predicted accurately with perfect recall
- Versicolor showed one error
- The 5 fold cross-validation conforms that I have an average accuracy of 96.67% conforming that the Logistic Regression model performs consistently across different subsets of iris.

Plot-

This plot show the classification report setosa, versicolor and virginica.



Summary-

- The logistic regression model classifies the iris species with high accuracy and perception of 96.67% showing strong performance.
- This report showed that setosa and virginica have perfect classification and versicolor have a very high classification.
- This is further verified using the 5-fold cross-validation where average accuracy is 96.67%.
- This project makes us understand how reliable is logistic regression for multi-class classification and has a strong foundation for more advanced machine learning applications.

Reference-

- Libraries - pandas, matplotlib, scikit-learn, statsmodels.
- Tools – Visual Studio Code