Data Analyst Internship Report

Introduction

Name	Vedant Maladkar
Project Title	E-commerce Furniture Dataset 2024
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
```

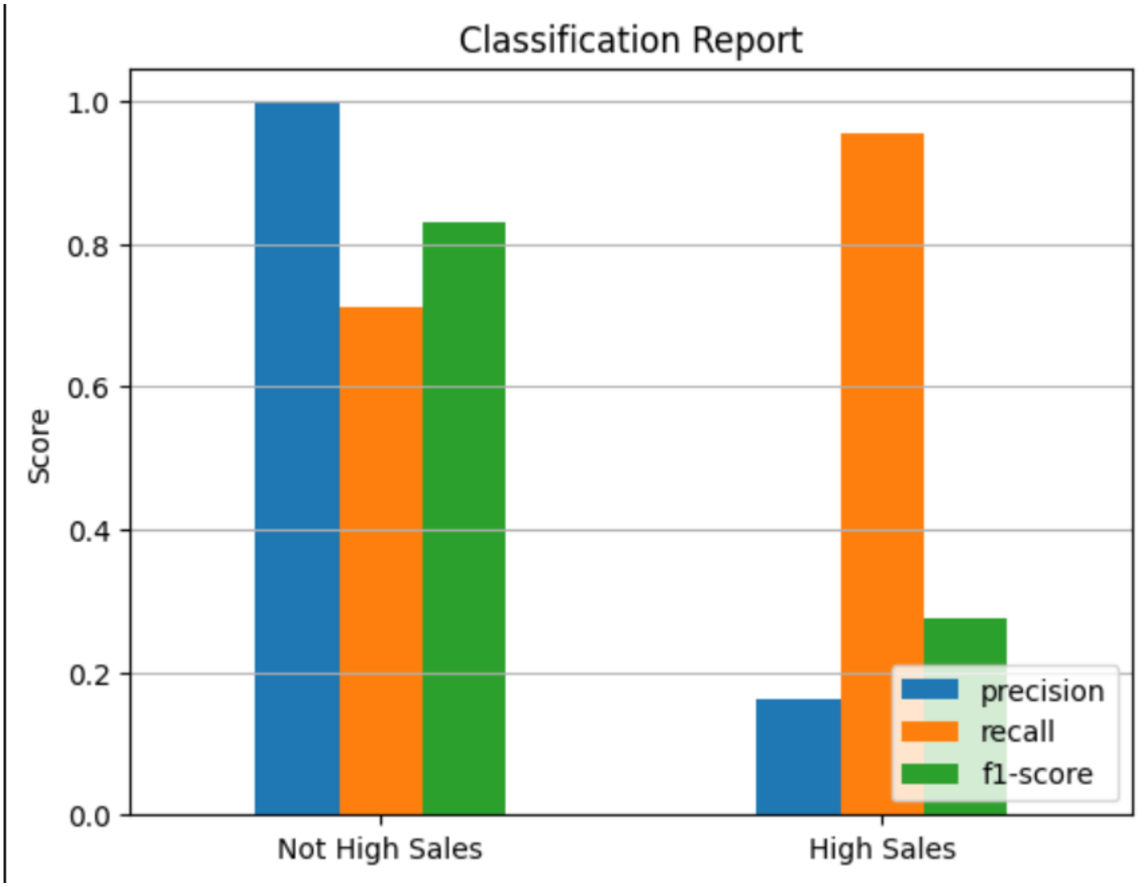
The Logistic Regression shows an accuracy score of approximately 73%

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