```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (accuracy_score, confusion_matrix,
                             classification_report, roc_auc_score,
                             precision_recall_curve,
average_precision_score)
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv(r'C:\Users\Rohit
Kadam\Downloads/supply_chain_data.csv')
df.head()
```

```
  Product type   SKU      Price  Availability  Number of products sold  \
0     haircare  SKU0  69.808006            55                      802

1     skincare  SKU1  14.843523            95                      736

2     haircare  SKU2  11.319683            34                        8

3     skincare  SKU3  61.163343            68                       83

4     skincare  SKU4   4.805496            26                      871


   Revenue generated Customer demographics  Stock levels  Lead
times  \
0        8661.996792            Non-binary            58           7

1        7460.900065                Female            53          30

2        9577.749626               Unknown             1          10

3        7766.836426            Non-binary            23          13

4        2686.505152            Non-binary             5           3
```

```
   Order quantities  ...  Location Lead time  Production volumes  \
0                96  ...    Mumbai        29                 215   
1                37  ...    Mumbai        23                 517   
2                88  ...    Mumbai        12                 971   
3                59  ...   Kolkata        24                 937   
4                56  ...     Delhi         5                 414   

  Manufacturing lead time Manufacturing costs  Inspection results  \
0                      29           46.279879             Pending   
1                      30           33.616769             Pending   
2                      27           30.688019             Pending   
3                      18           35.624741                Fail   
4                       3           92.065161                Fail   

   Defect rates  Transportation modes   Routes       Costs  
0      0.226410                  Road  Route B  187.752075  
1      4.854068                  Road  Route B  503.065579  
2      4.580593                   Air  Route C  141.920282  
3      4.746649                  Rail  Route A  254.776159  
4      3.145580                   Air  Route A  923.440632  

[5 rows x 24 columns]
```

```
df.shape
```

```
(100, 24)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Product type             100 non-null    object 
 1   SKU                      100 non-null    object 
 2   Price                    100 non-null    float64
 3   Availability             100 non-null    int64  
 4   Number of products sold  100 non-null    int64  
 5   Revenue generated        100 non-null    float64
 6   Customer demographics    100 non-null    object 
 7   Stock levels             100 non-null    int64  
 8   Lead times               100 non-null    int64  
 9   Order quantities         100 non-null    int64  
 10  Shipping times           100 non-null    int64  
 11  Shipping carriers        100 non-null    object 
 12  Shipping costs           100 non-null    float64
 13  Supplier name            100 non-null    object 
 14  Location                 100 non-null    object 
 15  Lead time                100 non-null    int64  
```

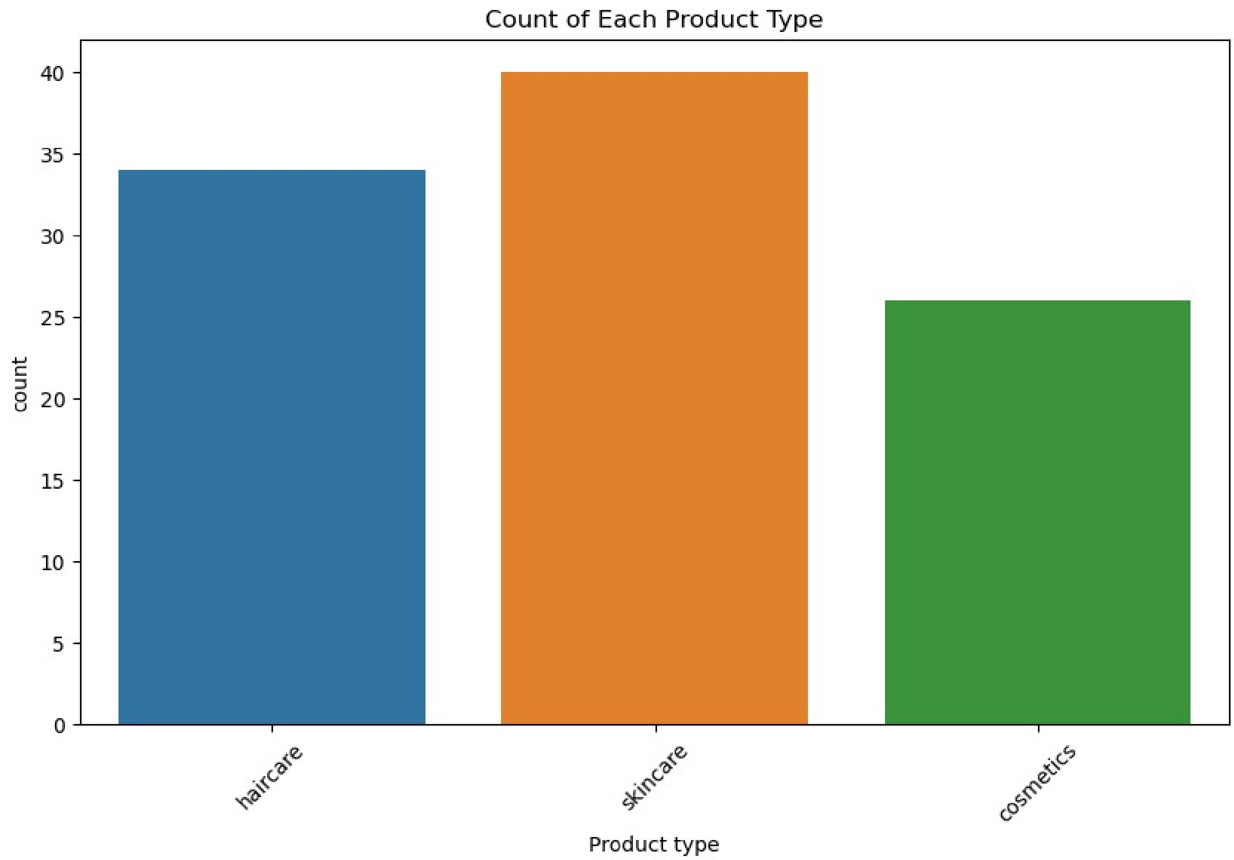
```
 16  Production volumes       100 non-null    int64
 17  Manufacturing lead time  100 non-null    int64
 18  Manufacturing costs      100 non-null    float64
 19  Inspection results       100 non-null    object
 20  Defect rates             100 non-null    float64
 21  Transportation modes     100 non-null    object
 22  Routes                   100 non-null    object
 23  Costs                    100 non-null    float64
dtypes: float64(6), int64(9), object(9)
memory usage: 18.9+ KB
```

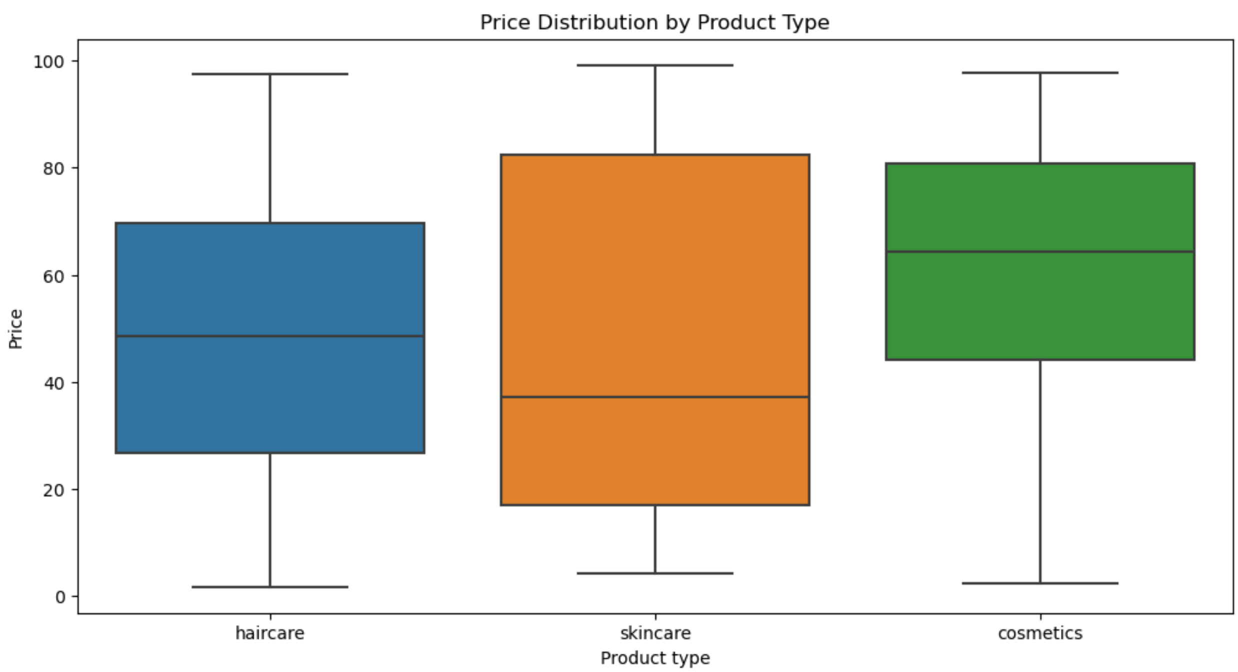
```
df.head()
```

```
  Product type   SKU      Price  Availability  Number of products sold
\
0     haircare  SKU0  69.808006            55                      802

1     skincare  SKU1  14.843523            95                      736

2     haircare  SKU2  11.319683            34                        8

3     skincare  SKU3  61.163343            68                       83

4     skincare  SKU4   4.805496            26                      871


   Revenue generated Customer demographics  Stock levels  Lead
times  \
0        8661.996792            Non-binary            58           7

1        7460.900065                Female            53          30

2        9577.749626               Unknown             1          10

3        7766.836426            Non-binary            23          13

4        2686.505152            Non-binary             5           3


   Order quantities  ...  Location Lead time  Production volumes  \
0                96  ...    Mumbai        29                 215
1                37  ...    Mumbai        23                 517
2                88  ...    Mumbai        12                 971
3                59  ...   Kolkata        24                 937
4                56  ...     Delhi         5                 414

  Manufacturing lead time Manufacturing costs  Inspection results  \
0                      29           46.279879             Pending
1                      30           33.616769             Pending
2                      27           30.688019             Pending
3                      18           35.624741                Fail
```

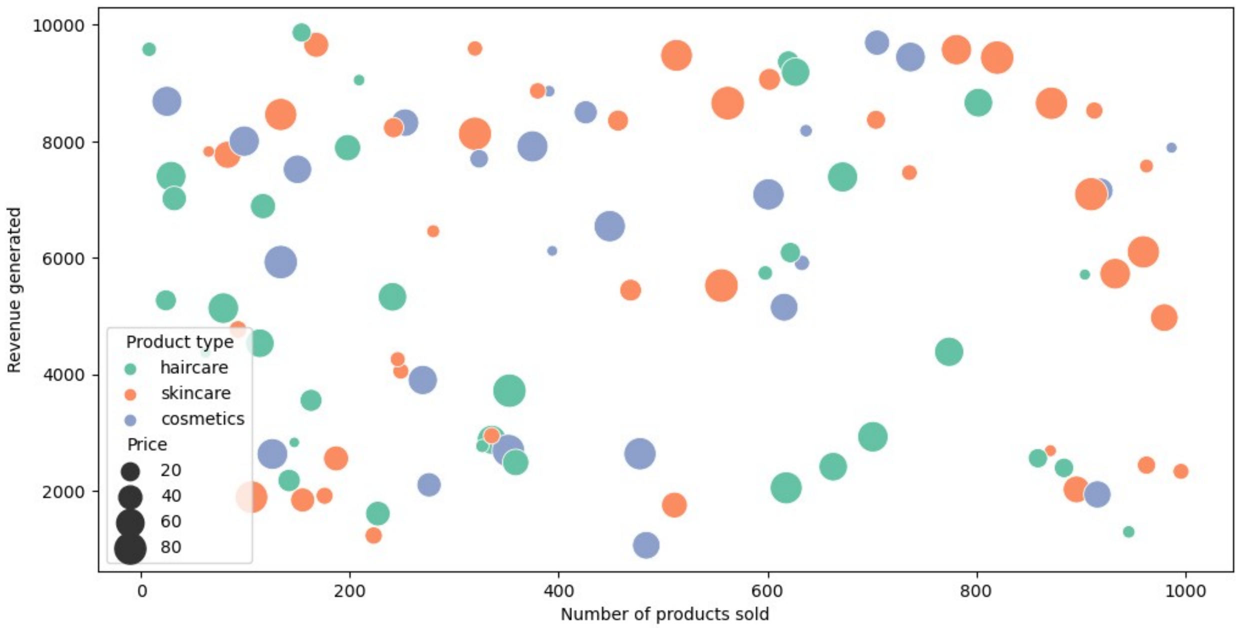
```
4                       3           92.065161                Fail

   Defect rates  Transportation modes   Routes       Costs
0      0.226410                  Road  Route B  187.752075
1      4.854068                  Road  Route B  503.065579
2      4.580593                   Air  Route C  141.920282
3      4.746649                  Rail  Route A  254.776159
4      3.145580                   Air  Route A  923.440632

[5 rows x 24 columns]
```

```
plt.figure(figsize=(10,6))
sns.countplot(data=df, x='Product type')
plt.title('Count of Each Product Type')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(data=df,x='Product type',y='Price')
plt.title('Price Distribution by Product Type')
plt.show()

plt.figure(figsize=(12, 6))
sns.scatterplot(data=df,
                x='Number of products sold',
                y='Revenue generated',
                hue='Product type',
                size='Price',
                palette='Set2',  # optional: you can change color 
palette
                sizes=(40, 400))
```



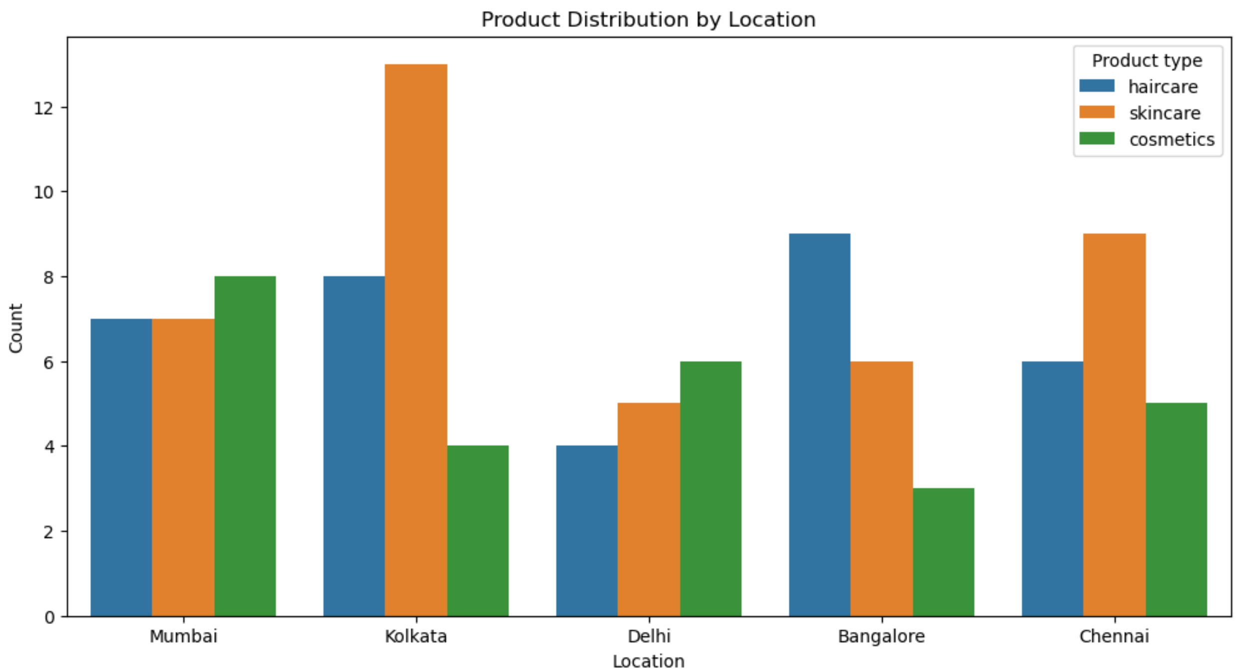



```
<Axes: xlabel='Number of products sold', ylabel='Revenue generated'>
```



2. Location and Supplier Analysis

```
# Location-wise product distribution
plt.figure(figsize=(12,6))
sns.countplot(data=df,x='Location',hue='Product type')
plt.title('Product Distribution by Location')
plt.ylabel('Count')
plt.show()
```



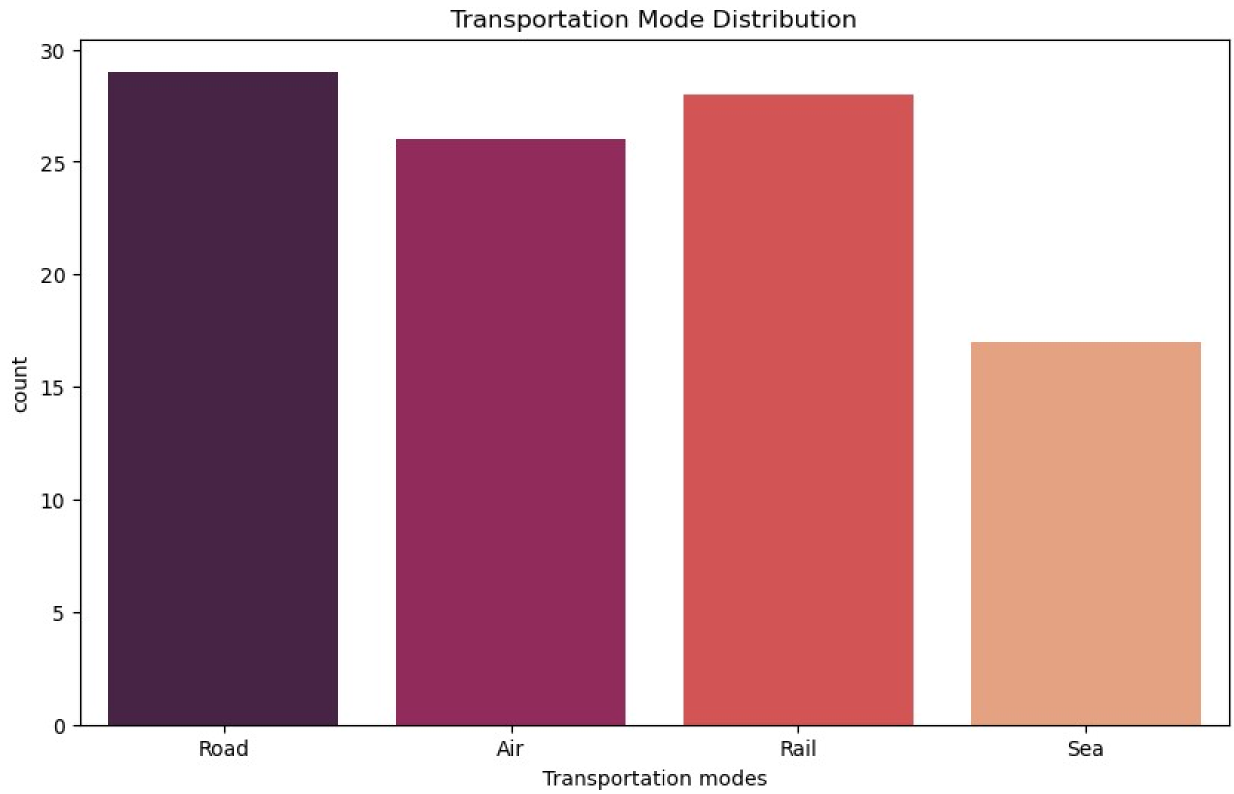
```
# Supplier performance metrics
supplier_metrics = df.groupby('Supplier name').agg({
    'Lead times': 'mean',
    'Manufacturing costs': 'mean',
    'Defect rates': 'mean'
}).reset_index()
supplier_metrics
```

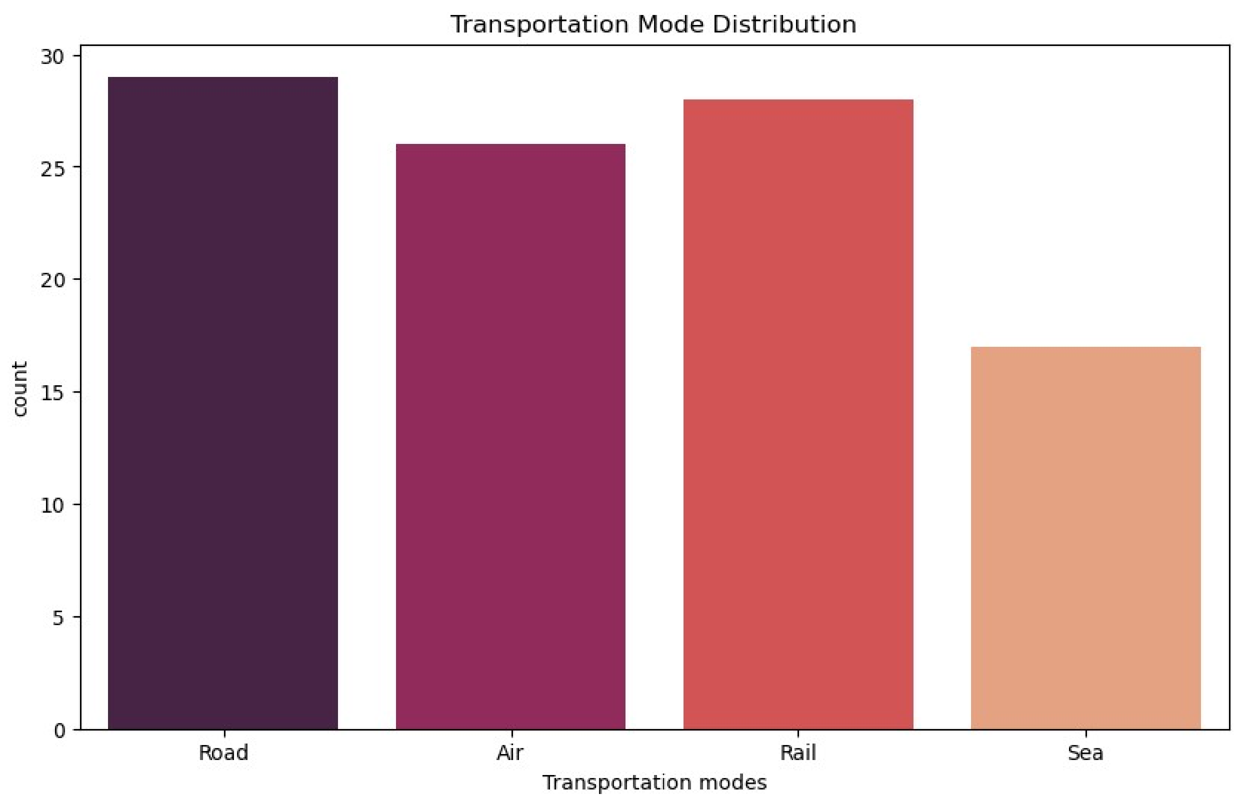
	Supplier name	Lead times	Manufacturing costs	Defect rates
0	Supplier 1	16.777778	45.254027	1.803630
1	Supplier 2	16.227273	41.622514	2.362750
2	Supplier 3	14.333333	43.634121	2.465786
3	Supplier 4	17.000000	62.709727	2.337397
4	Supplier 5	14.722222	44.768243	2.665408

```
fig = px.bar(supplier_metrics, x='Supplier name', y=['Lead times',
'Manufacturing costs', 'Defect rates'],
             title='Supplier Performance Metrics', barmode='group')
fig.show()
```

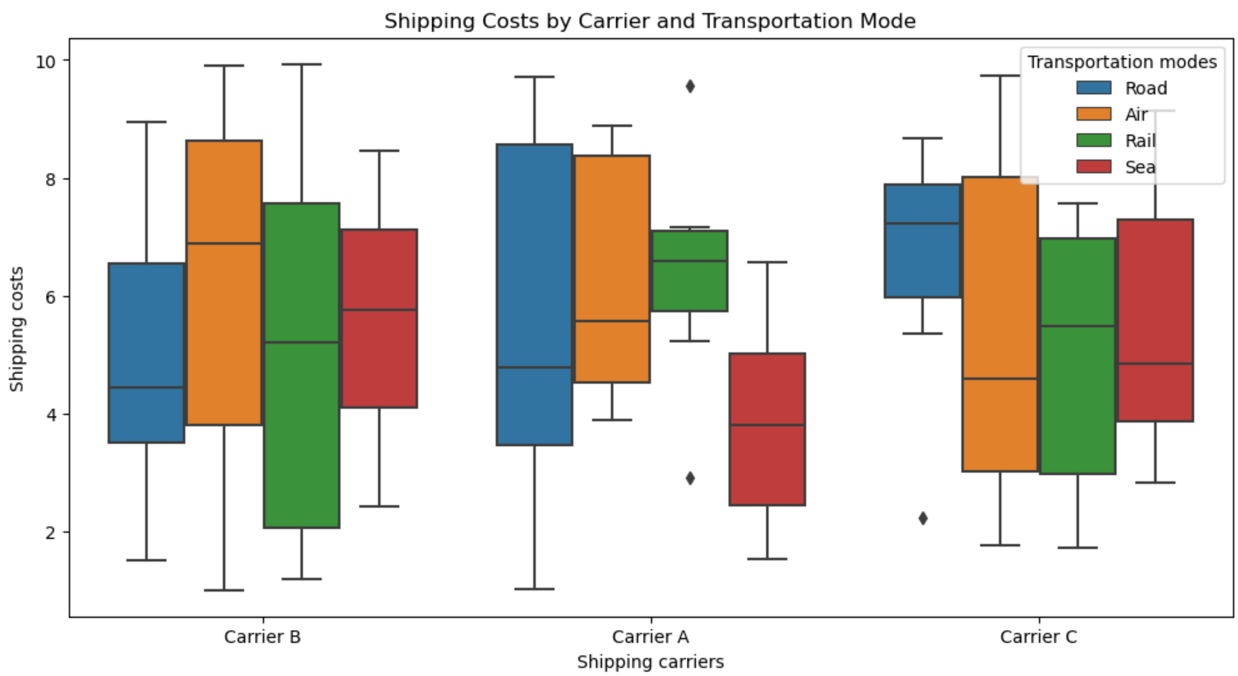
3. Transportation and Logistics Analysis

```
# Transportation mode analysis
plt.figure(figsize=(10, 6))
sns.countplot(x='Transportation modes', data=df, palette="rocket")
plt.title('Transportation Mode Distribution')
plt.show()
plt.figure(figsize=(10, 6))
sns.countplot(x='Transportation modes', data=df, palette="rocket")
plt.title('Transportation Mode Distribution')
plt.show()
```


Transportation Mode Distribution
count
30
25
20
15
10
5
0
Road
Air
Rail
Sea
Transportation modes


Transportation Mode Distribution
count
30
25
20
15
10
5
0
Road
Air
Rail
Sea
Transportation modes

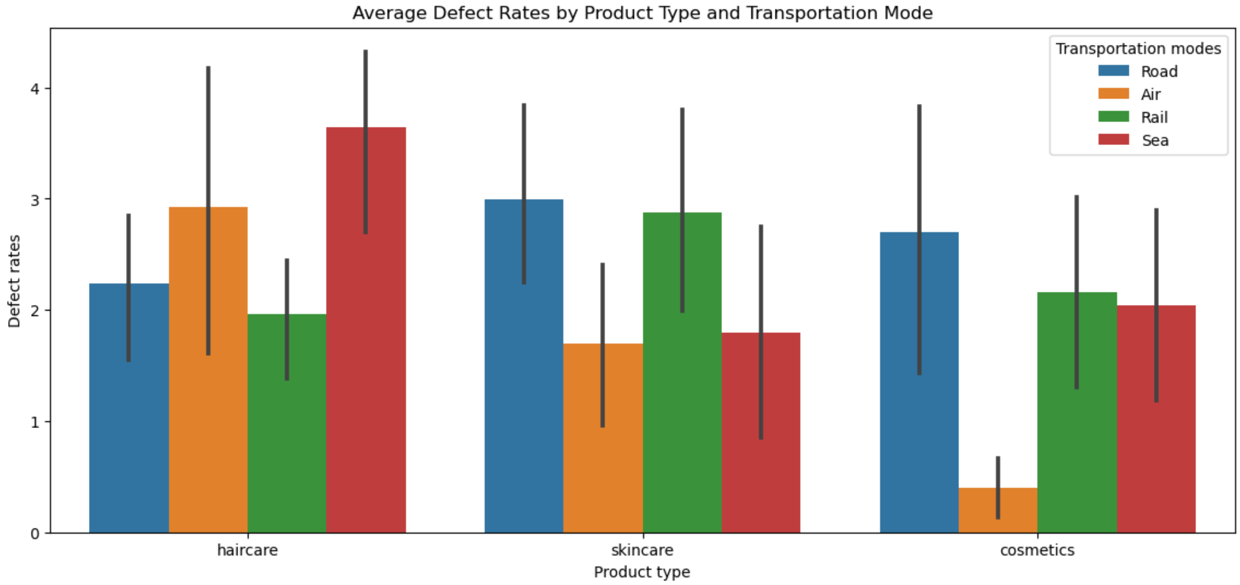
```
# Shipping costs by carrier
plt.figure(figsize=(12, 6))
sns.boxplot(x='Shipping carriers', y='Shipping costs',
hue='Transportation modes', data=df)
plt.title('Shipping Costs by Carrier and Transportation Mode')
plt.show()
```



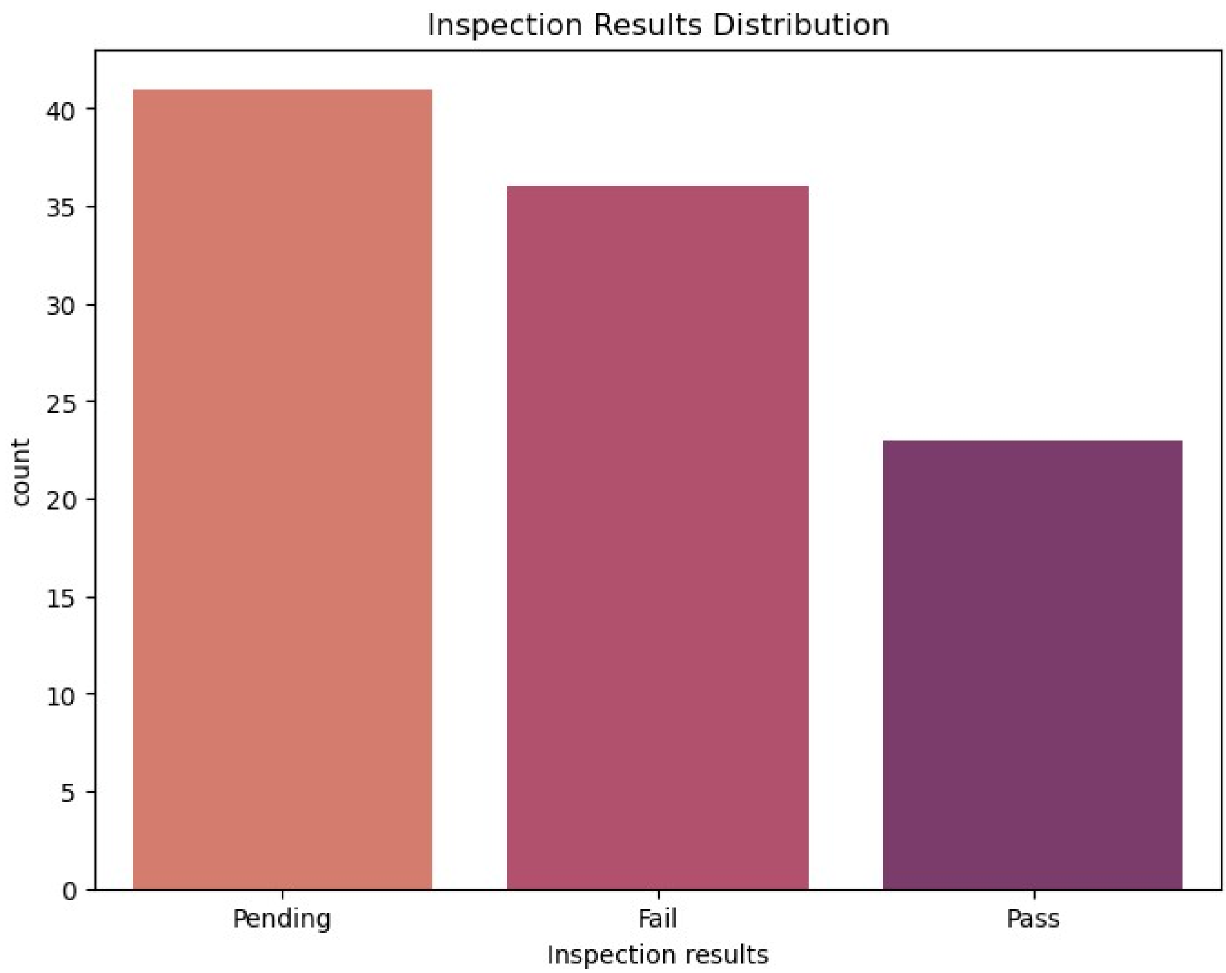

```
# Lead time vs Manufacturing costs
px.scatter(df, x='Lead times', y='Manufacturing costs',
                    color='Product type', size='Production volumes',
                    title='Lead Time vs Manufacturing Costs')
```

Quality Control Analysis

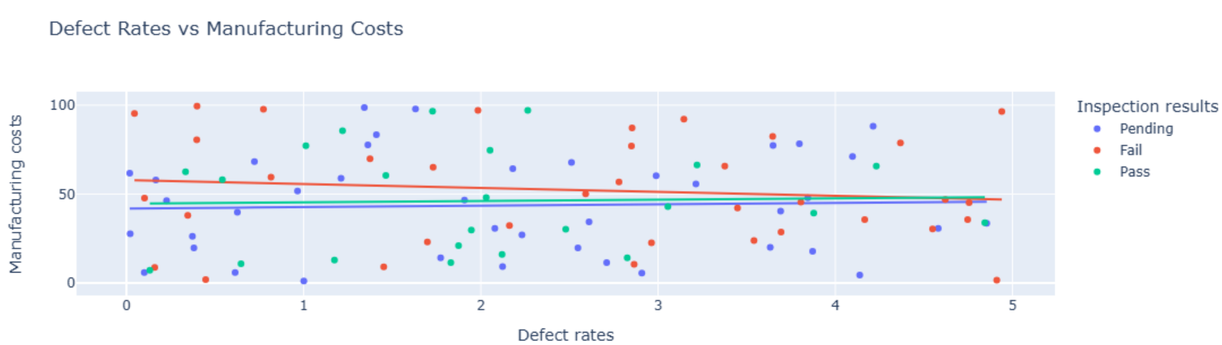
```
# Defect rates by product type and transportation mode
plt.figure(figsize=(14, 6))
sns.barplot(data=df,x='Product type',y='Defect
rates',hue='Transportation modes')
plt.title('Average Defect Rates by Product Type and Transportation
Mode')
plt.show()
```




```
# Inspection results
plt.figure(figsize=(8, 6))
sns.countplot(x='Inspection results', data=df, palette="flare")
plt.title('Inspection Results Distribution')
plt.show()
```




```python
# Defect rates vs Manufacturing costs
px.scatter(df, x='Defect rates', y='Manufacturing costs',
                      color='Inspection results', trendline="ols",
                      title='Defect Rates vs Manufacturing Costs')
```



Data Preprocessing for Predictive Modeling

```
# Create a target variable - High Defect Rate (1 if defect rate >
median, else 0)
median_defect = df['Defect rates'].median()
df['High_Defect'] = (df['Defect rates'] > median_defect).astype(int)

# Select features and target
features  = ['Price', 'Availability', 'Number of products sold',
'Revenue generated',
       'Stock levels', 'Lead times', 'Order quantities', 'Shipping
times',
       'Shipping costs', 'Lead time', 'Production volumes',
       'Manufacturing lead time', 'Manufacturing costs']
target = 'High_Defect'

X = df[features]
y = df[target]

# Split data into train and test sets
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.3,random_state=42,stratify=y)

# Scale numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

Predictive Modeling

```
# Initialize and train logistic regression
lr = LogisticRegression(max_iter=1000, random_state=42)
lr.fit(X_train_scaled, y_train)

# Make predictions
y_pred_lr = lr.predict(X_test_scaled)
y_prob_lr = lr.predict_proba(X_test_scaled)[:, 1]

# Evaluate model
print("Logistic Regression Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_lr):.4f}")
print(f"ROC AUC: {roc_auc_score(y_test, y_prob_lr):.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred_lr))
```

```
Logistic Regression Performance:
Accuracy: 0.6333
ROC AUC: 0.6622
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.67      0.65        15
           1       0.64      0.60      0.62        15

    accuracy                           0.63        30
   macro avg       0.63      0.63      0.63        30
weighted avg       0.63      0.63      0.63        30
```

2. Random Forest Classifier

```python
# Initialize and train random forest
rf = RandomForestClassifier(n_estimators=200, max_depth=10,
                            random_state=42, n_jobs=-1)
rf.fit(X_train_scaled, y_train)

# Make predictions
y_pred_rf = rf.predict(X_test_scaled)
y_prob_rf = rf.predict_proba(X_test_scaled)[:, 1]

# Evaluate model
print("Random Forest Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")
print(f"ROC AUC: {roc_auc_score(y_test, y_prob_rf):.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Performance:
Accuracy: 0.4667
ROC AUC: 0.5044

Classification Report:
              precision    recall  f1-score   support

           0       0.46      0.40      0.43        15
           1       0.47      0.53      0.50        15

    accuracy                           0.47        30
   macro avg       0.47      0.47      0.46        30
weighted avg       0.47      0.47      0.46        30
```