Insurance || CAP4770 Final Project

Matthew Bajdas and Maddy Wirbel

In [8]:
```
# load libraries needed for project here <----
import pandas as pd
import numpy as np
import seaborn as sns
import re
import shap
import matplotlib.pyplot as plt
from prefixspan import PrefixSpan
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, fpgrowth, association_rules
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.preprocessing import StandardScaler, LabelEncoder, MinMaxScaler
from sklearn.metrics import confusion_matrix, roc_curve, auc, classification_report
from sklearn.cluster import KMeans, DBSCAN
from sklearn.ensemble import IsolationForest
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Task 1: Data Preprocessing

In [9]:
```
df = pd.read_csv("insuranceClaims.csv") # initialize data frame

# handle non-numeric errors in numeric columns
df['vehicle_age'] = pd.to_numeric(df['vehicle_age'], errors='coerce')
df['customer_age'] = pd.to_numeric(df['customer_age'], errors='coerce')

# handle missing values
df.fillna({
    'customer_age': df['customer_age'].median(),
    'vehicle_age': df['vehicle_age'].median(),
    'region_code': df['region_code'].mode()[0],
}, inplace = True)

# creating binned data
df['vehicle_age_binned'] = pd.cut(df['vehicle_age'], bins=[0,5,10,float('inf')], la
df['region_density_binned'] = pd.cut(df['region_density'], bins=[0,30,70,float('inf
df['airbags_binned'] = df['airbags'].map(
    lambda x: 'No Airbags' if x == 0 else ('1 Airbag' if x == 1 else '2+ Airbags')
)
df['claim_status_binned'] = df['claim_status'].map({0: 'No_Claim', 1: 'Claim'})
df['fuel_type_binned'] = df['fuel_type'].apply(lambda x: 'Fuel' if x in ['Diesel',
df['subscription_length_orig'] = df['subscription_length']

# normalize data
# ---> normalizing fields that are continuous
```

```
scaler = StandardScaler()
df[['subscription_length', 'customer_age', 'vehicle_age', 'turning_radius', 'region
    df[['subscription_length', 'customer_age', 'vehicle_age', 'turning_radius', 're

# horsepower normalization
def parse_bhp_rpm(value):
    match = re.match(r"(\d+(\.\d+)?)bhp@(\d+)rpm", value)
    if match:
        bhp = float(match.group(1))
        rpm = int(match.group(3))
        return bhp, rpm
    return None, None # else

bhp_rpm = [parse_bhp_rpm(entry) for entry in df['max_power']]
df[['BHP', 'RPM']] = pd.DataFrame(bhp_rpm, columns=['BHP', 'RPM'])

scaler = MinMaxScaler() # normalize w/ scaler
df[['BHP_normal', 'RPM_normal']] = scaler.fit_transform(df[['BHP', 'RPM']])

# encode categorical variables
# TODO: any other important categorical variables encode here <----
encoder = LabelEncoder()
df['region_code'] = encoder.fit_transform(df['region_code'])
df['transmission_type'] = encoder.fit_transform(df['transmission_type'])
df['steering_type'] = encoder.fit_transform(df['steering_type'])
df['is_esc'] = encoder.fit_transform(df['is_esc'])
```

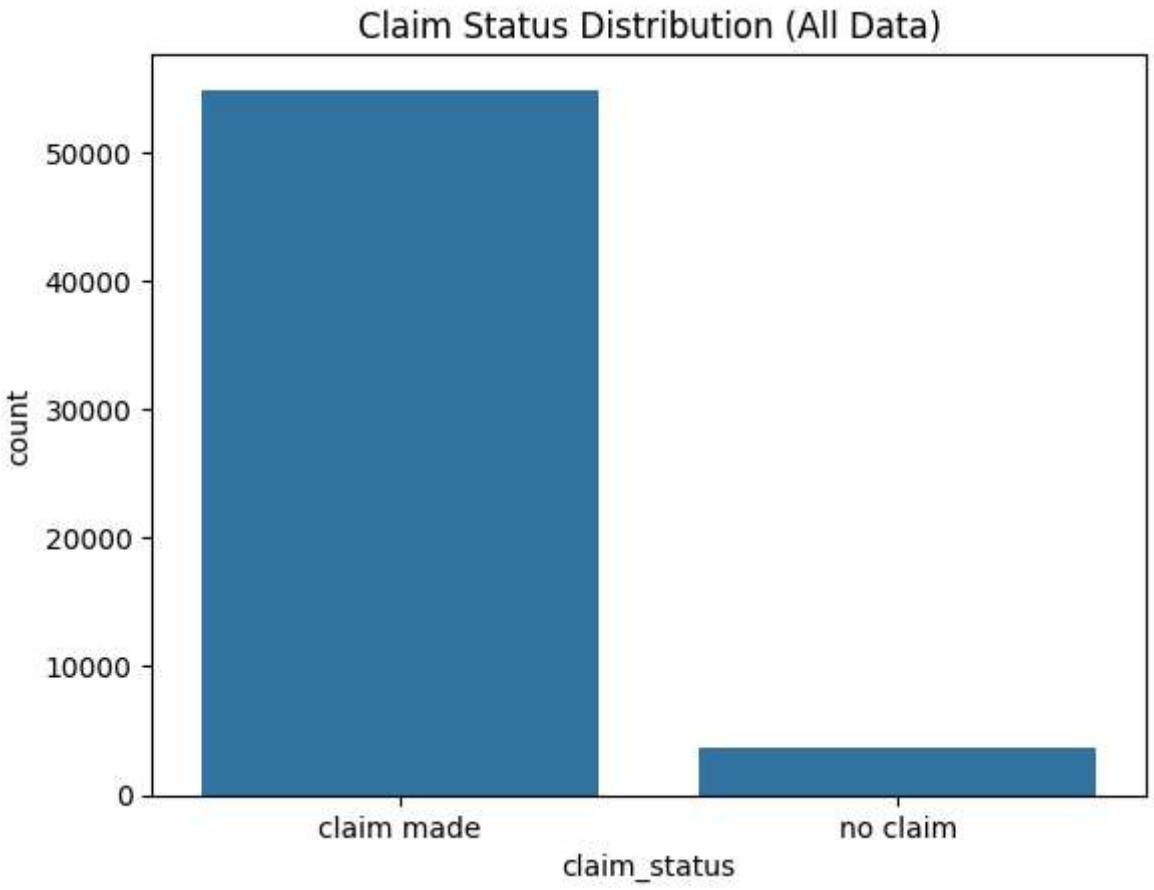
In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58592 entries, 0 to 58591
Data columns (total 51 columns):
 #   Column                              Non-Null Count  Dtype   
---  ------                              --------------  -----   
 0   policy_id                           58592 non-null  object  
 1   subscription_length                 58592 non-null  float64 
 2   vehicle_age                         58592 non-null  float64 
 3   customer_age                        58592 non-null  float64 
 4   region_code                         58592 non-null  int64   
 5   region_density                      58592 non-null  float64 
 6   segment                             58592 non-null  object  
 7   model                               58592 non-null  object  
 8   fuel_type                           58592 non-null  object  
 9   max_torque                          58592 non-null  object  
 10  max_power                           58592 non-null  object  
 11  engine_type                         58592 non-null  object  
 12  airbags                             58592 non-null  int64   
 13  is_esc                              58592 non-null  int64   
 14  is_adjustable_steering              58592 non-null  object  
 15  is_tpms                             58592 non-null  object  
 16  is_parking_sensors                  58592 non-null  object  
 17  is_parking_camera                   58592 non-null  object  
 18  rear_brakes_type                    58592 non-null  object  
 19  displacement                        58592 non-null  int64   
 20  cylinder                            58592 non-null  int64   
 21  transmission_type                   58592 non-null  int64   
 22  steering_type                       58592 non-null  int64   
 23  turning_radius                      58592 non-null  float64 
 24  length                              58592 non-null  int64   
 25  width                               58592 non-null  int64   
 26  gross_weight                        58592 non-null  int64   
 27  is_front_fog_lights                 58592 non-null  object  
 28  is_rear_window_wiper                58592 non-null  object  
 29  is_rear_window_washer               58592 non-null  object  
 30  is_rear_window_defogger             58592 non-null  object  
 31  is_brake_assist                     58592 non-null  object  
 32  is_power_door_locks                 58592 non-null  object  
 33  is_central_locking                  58592 non-null  object  
 34  is_power_steering                   58592 non-null  object  
 35  is_driver_seat_height_adjustable    58592 non-null  object  
 36  is_day_night_rear_view_mirror       58592 non-null  object  
 37  is_ecw                              58592 non-null  object  
 38  is_speed_alert                      58592 non-null  object  
 39  ncap_rating                         58592 non-null  float64 
 40  claim_status                        58592 non-null  int64   
 41  vehicle_age_binned                  53335 non-null  category
 42  region_density_binned               58592 non-null  category
 43  airbags_binned                      58592 non-null  object  
 44  claim_status_binned                 58592 non-null  object  
 45  fuel_type_binned                    58592 non-null  object  
 46  subscription_length_orig            58592 non-null  float64 
 47  BHP                                 58592 non-null  float64 
 48  RPM                                 58592 non-null  int64   
 49  BHP_normal                          58592 non-null  float64 
 50  RPM_normal                          58592 non-null  float64 
```

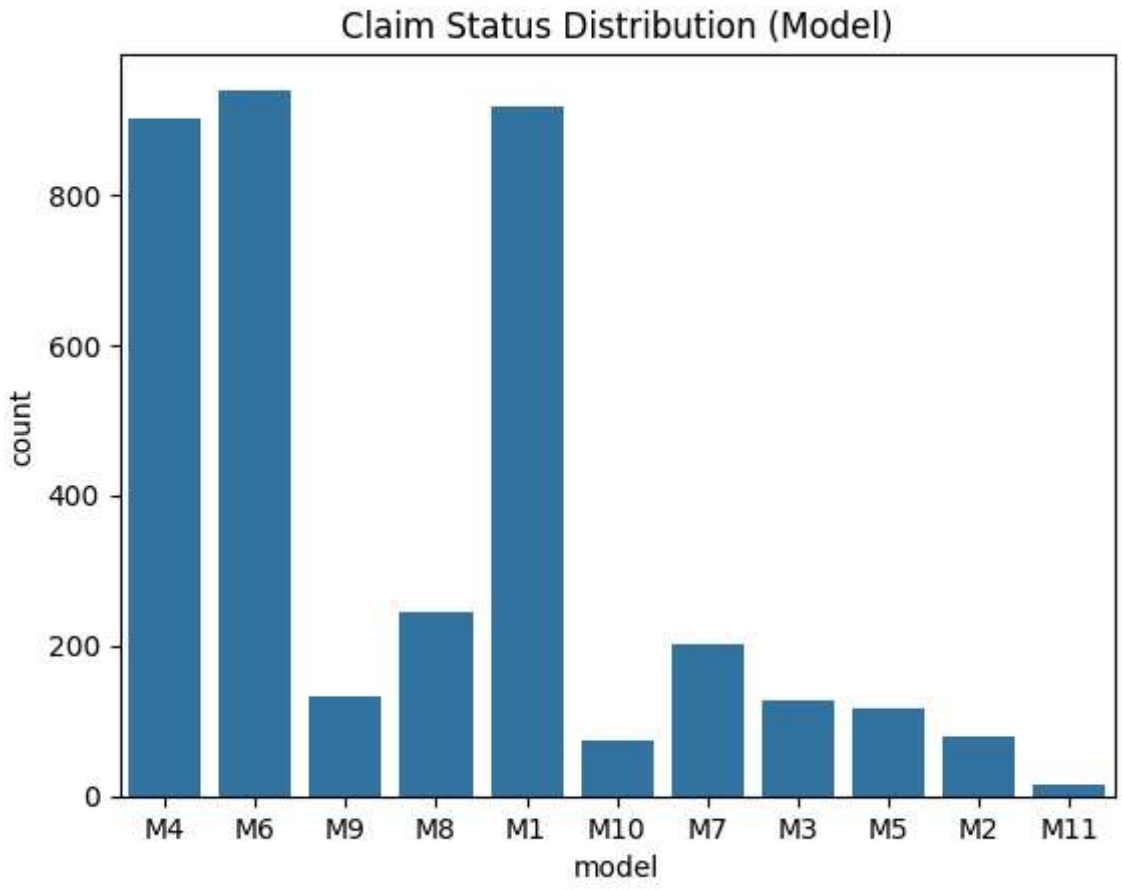
```
dtypes: category(2), float64(10), int64(12), object(27)
memory usage: 22.0+ MB
```

Task 2: Exploratory Data Analysis (EDA)

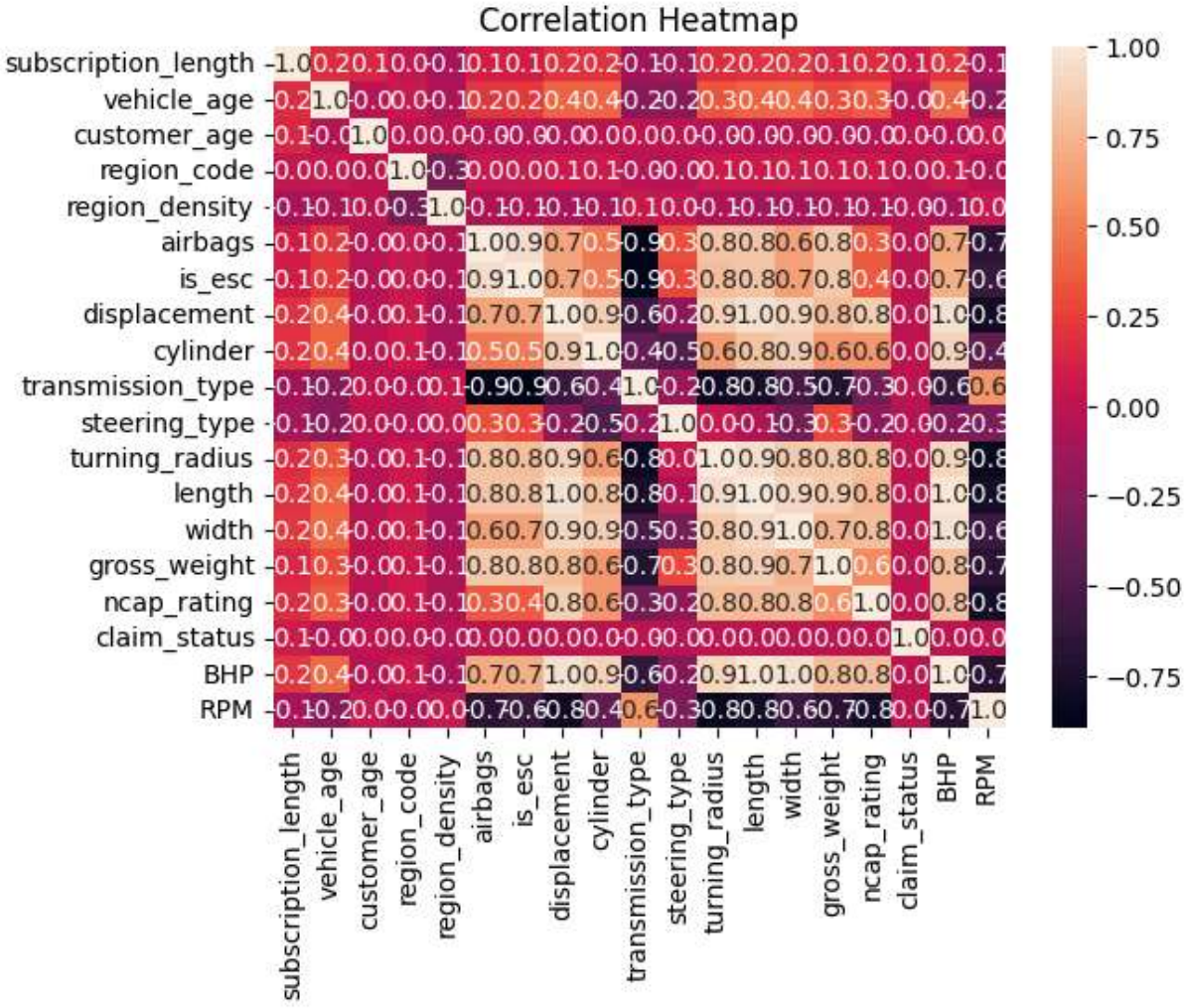
In [11]:
```
# claims distribution analysis -- varies across vehicles and demographics
sns.countplot(x = 'claim_status', data = df)
plt.title('Claim Status Distribution (All Data)')
plt.xticks([0, 1], {'no claim', 'claim made'})
plt.show()
```



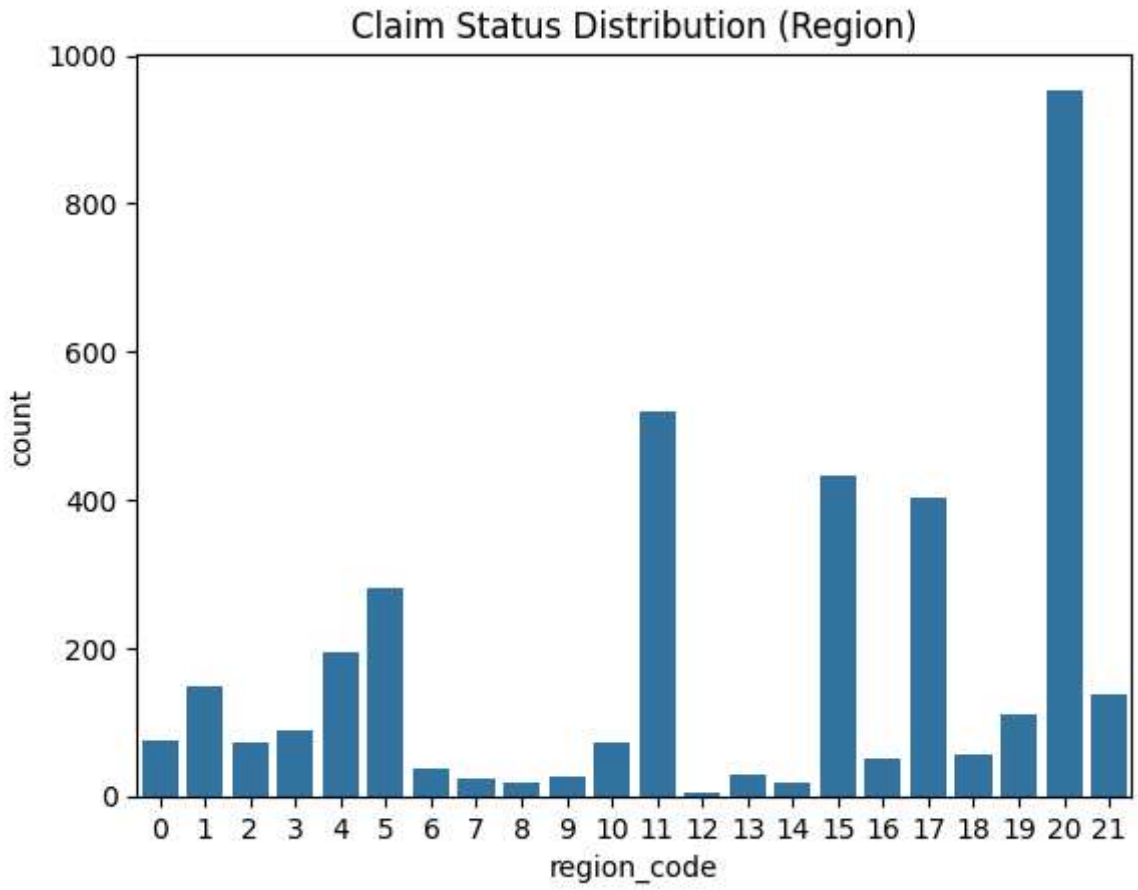
In [12]:
```
# additional claim status distribution --> vehicle models
df_claim_model = df[['claim_status', 'model']] # subset df
df_claim_only_model = df_claim_model[df_claim_model['claim_status'] == 1] # made a

sns.countplot(x = 'model', data = df_claim_only_model)
plt.title('Claim Status Distribution (Model)')
plt.show()
```



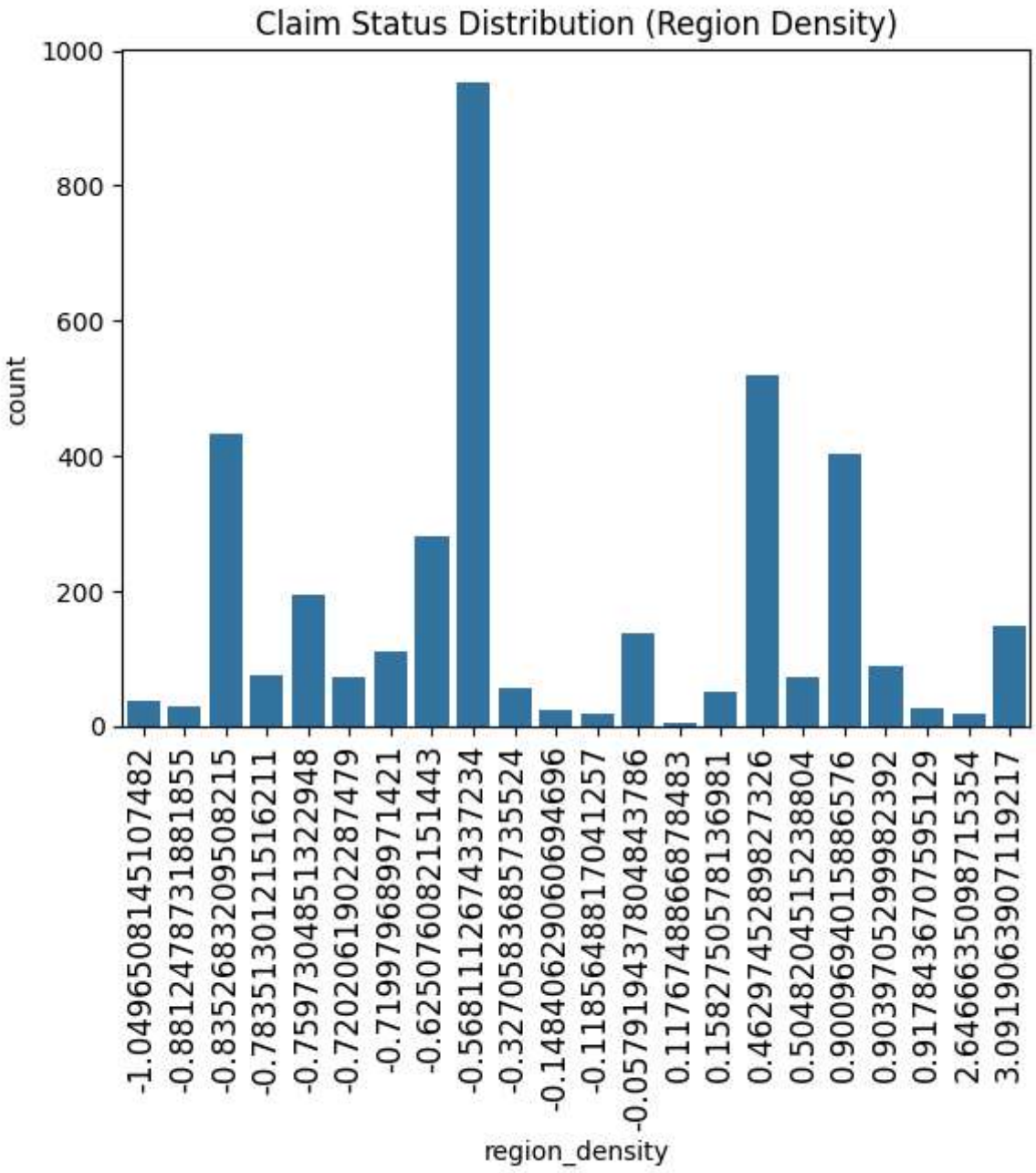
In [13]:
```
# correlation analysis -- relationship between contiguous feat. and claim_status
# use heatmaps
numeric_df = df.select_dtypes(include = [np.number])
numeric_df = numeric_df.drop(columns=numeric_df[['BHP_normal', 'RPM_normal', 'subsc

sns.heatmap(numeric_df.corr(), annot = True, fmt = '0.1f')
plt.title('Correlation Heatmap')
plt.show()
```



In [14]:
```
# regional claim analysis -- examine region_code and region_density
df_claim_region = df[['claim_status', 'region_code']] # subset df
df_claim_only_region = df_claim_region[df_claim_region['claim_status'] == 1] # made

sns.countplot(x = 'region_code', data = df_claim_only_region)
plt.title('Claim Status Distribution (Region)')
plt.show()
```



In [15]:
```
# additional regional claim analysis -- region_density
df_claim_density = df[['claim_status', 'region_density']] # subset df
df_claim_only_density = df_claim_density[df_claim_density['claim_status'] == 1] # m

sns.countplot(x = 'region_density', data = df_claim_only_density)
plt.title('Claim Status Distribution (Region Density)')
plt.xticks(rotation = 90, fontsize = 12)
plt.show()
```



Risk Segmentation

Task 1: Customer Segmentation

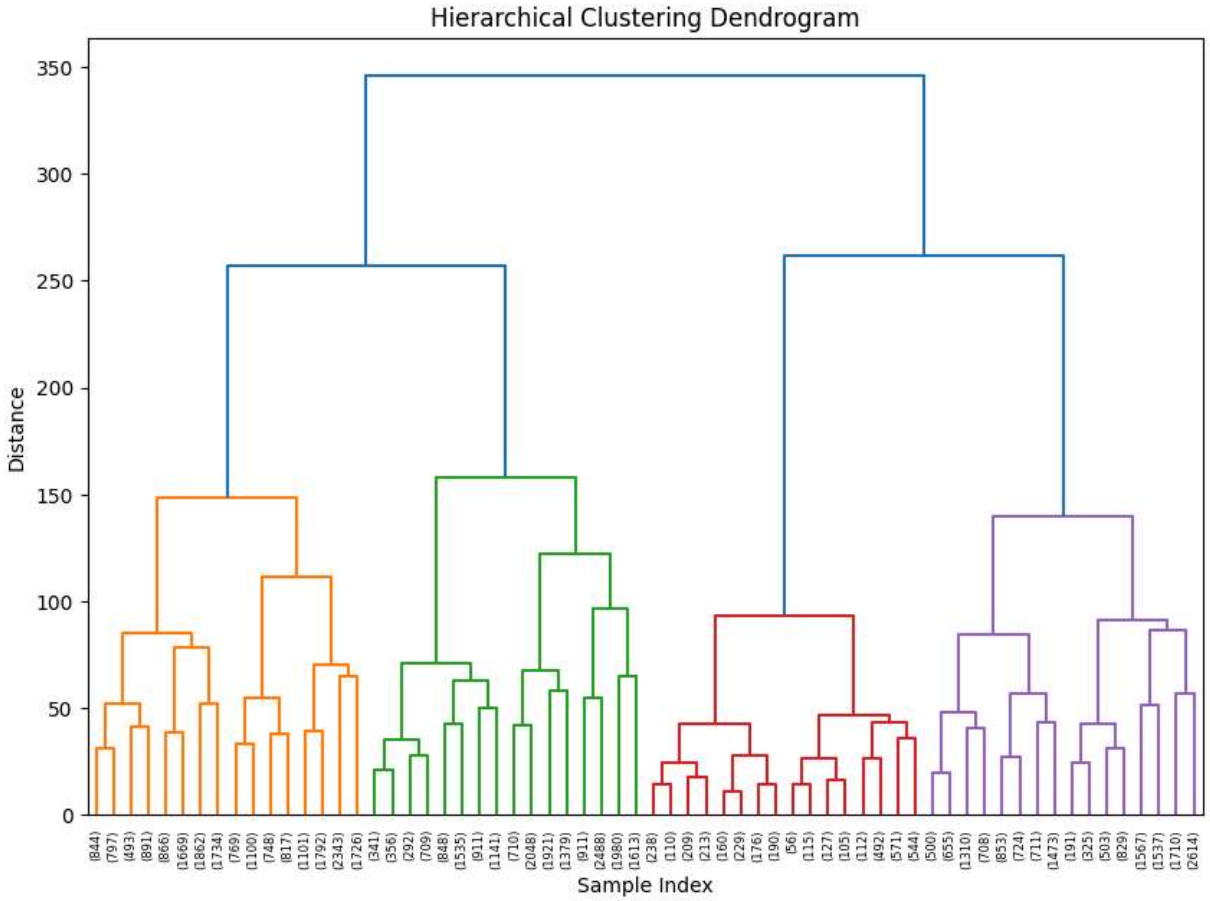
In [16]:
```
# K-means clustering
cluster_features = ['vehicle_age', 'customer_age', 'region_density', 'BHP_normal',
kmeans = KMeans(n_clusters = 4, random_state = 42)
df['kmeans_risk_cluster'] = kmeans.fit_predict(df[cluster_features])

pca = PCA(n_components=2)
pca_result = pca.fit_transform(df[cluster_features])

plt.scatter(pca_result[:, 0], pca_result[:, 1], c=df['kmeans_risk_cluster'], cmap='
plt.title('Customer Segmentation')
```

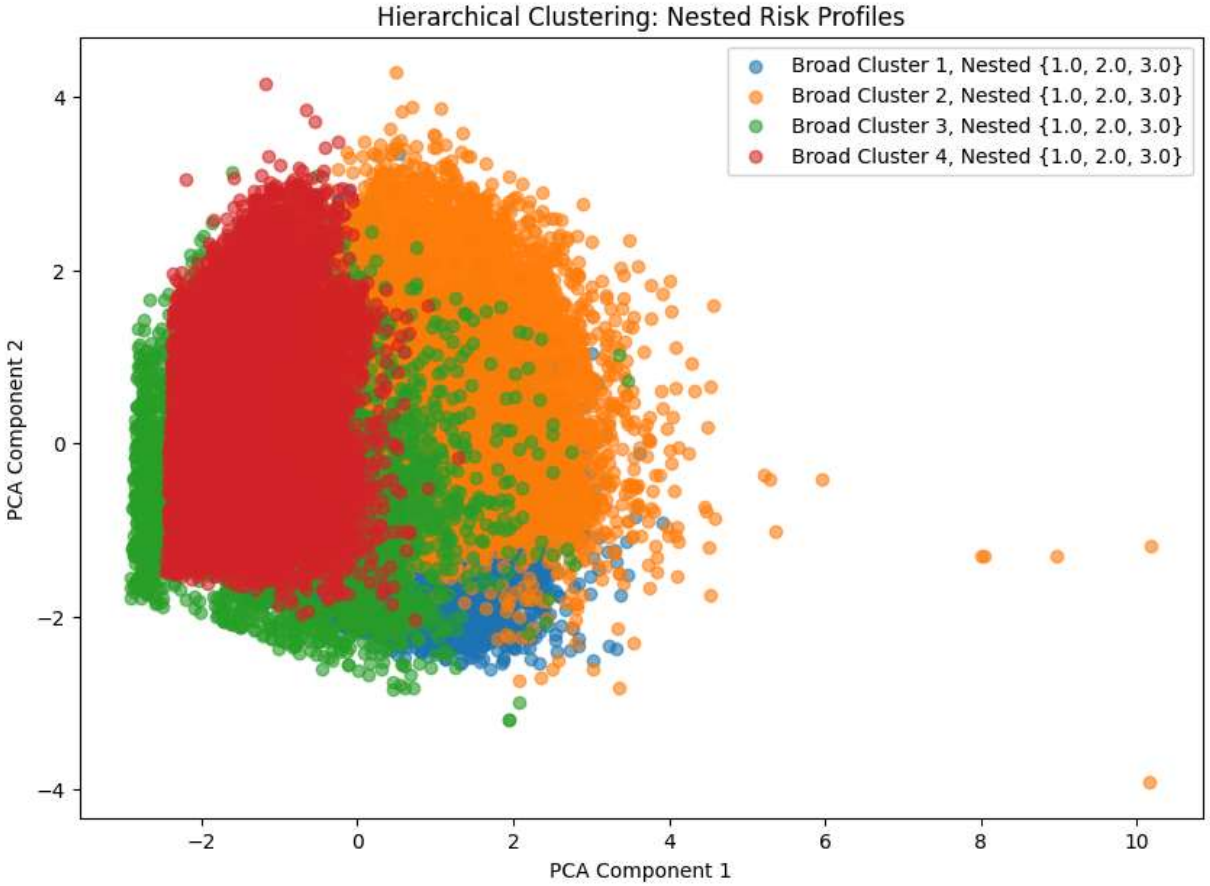
```
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()

cluster_summary = df.groupby('kmeans_risk_cluster')[cluster_features].mean()
print(cluster_summary)
```

kmeans_risk_cluster	vehicle_age	customer_age	region_density	BHP_normal	RPM_normal	ncap_rating	subscription_length
0	-0.718508	0.081307	-0.148811	0.123299	0.991134	-1.139589	-0.389025
1	0.372242	0.256937	-0.263223	0.675175	0.599677	0.496598	1.059726
2	0.336428	-0.390557	-0.180808	0.686225	0.541782	0.636744	-0.795322
3	-0.195744	0.000030	3.032052	0.384010	0.781527	-0.268910	-0.328549

In [17]:
```
# ***** This code section will run for a couple minutes *****
# hierarchial clustering
cluster_features = ['vehicle_age', 'customer_age', 'region_density', 'BHP_normal',
linkage_matrix = linkage(df[cluster_features], method='ward')

plt.figure(figsize=(10,7))
dendrogram(linkage_matrix, truncate_mode='level', p=5)
```

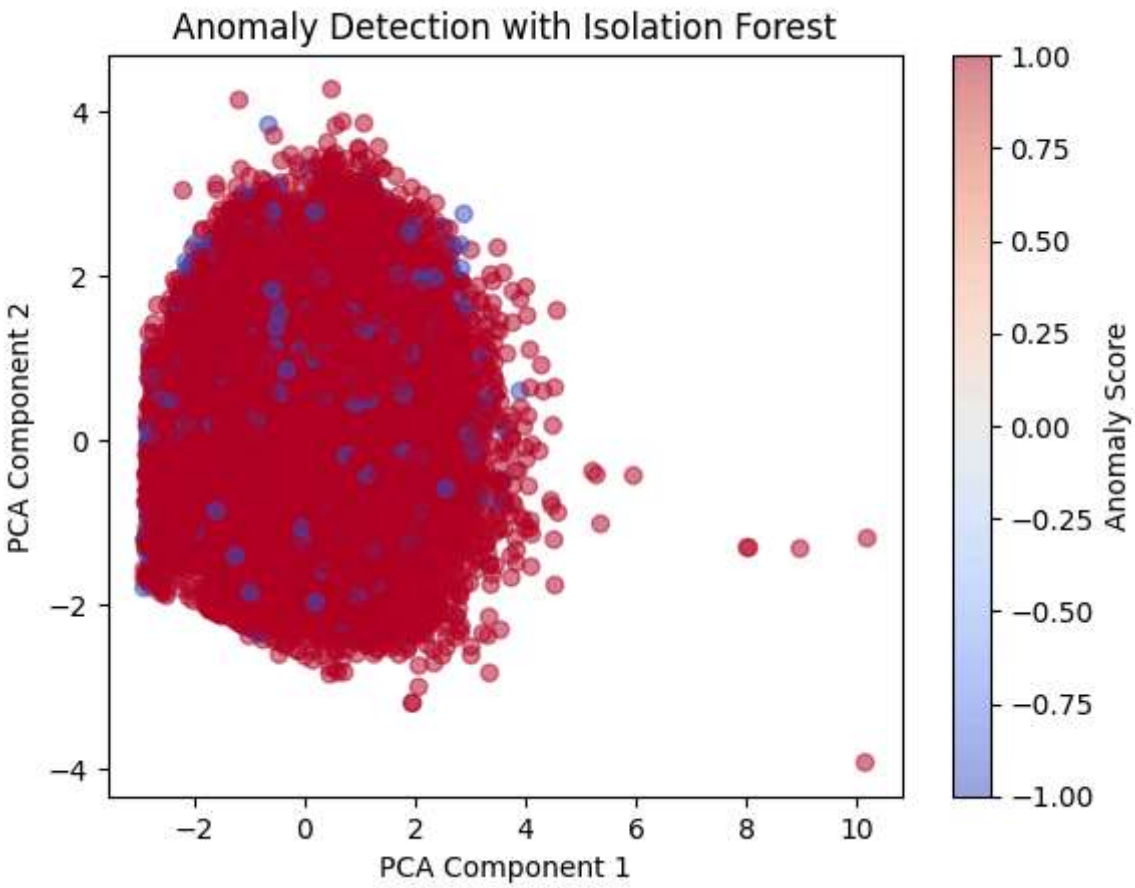
```
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()

df['broad_cluster'] = fcluster(linkage_matrix, t=4, criterion='maxclust')
for cluster in df['broad_cluster'].unique():
    cluster_data = df[df['broad_cluster'] == cluster]

    clustering_features = df[cluster_features][df['broad_cluster'] == cluster]
    cluster_linkage = linkage(clustering_features, method='ward')

    df.loc[df['broad_cluster'] == cluster, 'nested_cluster'] = fcluster(cluster_lin

pca = PCA(n_components=2)
pca_result = pca.fit_transform(df[cluster_features])

plt.figure(figsize=(10,7))
for broad_cluster in df['broad_cluster'].unique():
    cluster_data = df[df['broad_cluster'] == broad_cluster]
    nested_clusters = set(cluster_data['nested_cluster'])
    plt.scatter(
        pca_result[cluster_data.index, 0],
        pca_result[cluster_data.index, 1],
        label=f'Broad Cluster {broad_cluster}, Nested {nested_clusters}',
        alpha=0.6
    )

plt.title('Hierarchical Clustering: Nested Risk Profiles')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.show()
```





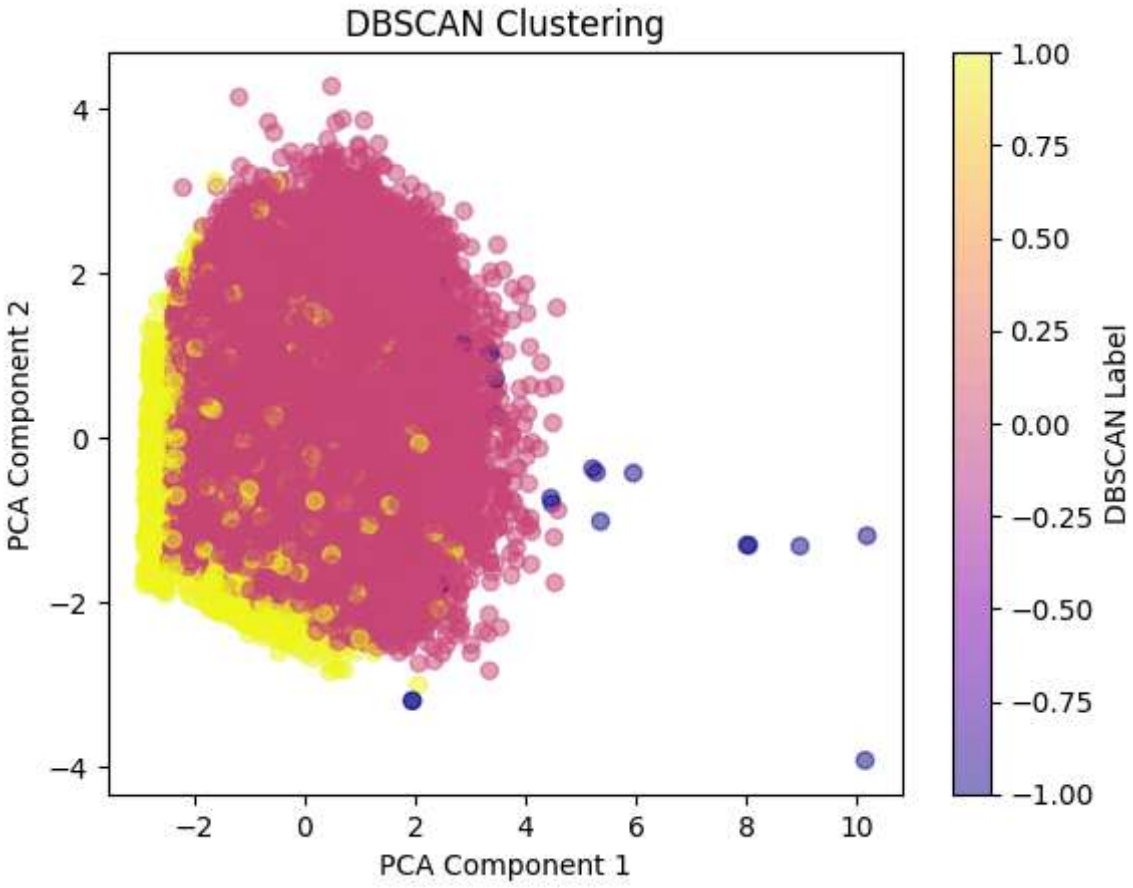
Task 2: Anomaly Detection

In [18]:
```
# isolation forest
iso_sub_df = df[['BHP_normal', 'claim_status', 'is_esc']]

iso_sub_df = iso_sub_df.fillna(iso_sub_df.mean())

# values of BHP and claim status are already normalized

iso_df = pd.DataFrame(iso_sub_df, columns = ['BHP', 'claim_status'])
iso_df['is_esc'] = df['is_esc']
iso_df['BHP'] = df['BHP_normal']

iso_forest = IsolationForest(contamination=0.05, random_state=42) # detecting for 5
iso_forest.fit(iso_df)

df['anomaly'] = iso_forest.predict(iso_df)
unusual_patterns = df[df['anomaly'] == -1] # take out

plt.scatter(pca_result[:, 0], pca_result[:, 1], c=df['anomaly'], cmap='coolwarm', a
plt.title('Anomaly Detection with Isolation Forest')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Anomaly Score')
plt.show()

print("Unusual vehicle-claim patterns detected:")
print(unusual_patterns[['policy_id', 'BHP_normal', 'is_esc', 'claim_status']])
```



```
Unusual vehicle-claim patterns detected:
       policy_id  BHP_normal  is_esc  claim_status
12     POL050280    0.937051       1             1
76     POL044165    0.737564       0             1
81     POL012008    0.199487       0             1
84     POL043686    0.000000       0             1
103    POL036840    0.000000       0             1
...          ...         ...     ...           ...
58463  POL032323    0.937051       1             1
58475  POL052302    0.937051       1             1
58517  POL053069    0.617179       1             1
58524  POL039299    0.000000       0             1
58548  POL058560    0.617179       1             1

[2809 rows x 4 columns]
```

In [19]:
```
# DBSCAN
dbscan = DBSCAN(eps=1.5, min_samples=10)
df['dbscan_label'] = dbscan.fit_predict(df[cluster_features])

plt.scatter(pca_result[:, 0], pca_result[:, 1], c=df['dbscan_label'], cmap='plasma'
plt.title('DBSCAN Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='DBSCAN Label')
plt.show()

dbscan_summary = df['dbscan_label'].value_counts()
print(dbscan_summary)
```




```
dbscan_label
 0    54931
 1     3638
-1       23
Name: count, dtype: int64
```

Predictive Modeling

Task 1: Classification Model

In [20]:
```
non_numeric_cols = df.select_dtypes(include=['object', 'category']).columns
x = df.drop(columns=non_numeric_cols)
x = x.drop(columns='claim_status')
y = df['claim_status']
y = y.astype('float32')
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta
```

In [21]:
```
# neural network
model = Sequential([
    Dense(32, activation = 'relu', input_dim = x_train.shape[1]),
    Dense(16, activation = 'relu'),
    Dense(1, activation = 'sigmoid')
])
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accurac
model.fit(x_train, y_train, validation_split = 0.2, epochs = 20, batch_size = 32)

# evaluation of neural network
```

```
y_pred = (model.predict(x_test) > 0.5).astype(int)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Epoch 1/20
```

```
c:\Users\mtbaj\Documents\vscode-workspace\.venv\Lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8866 - loss: 17.3339 - val_
accuracy: 0.8781 - val_loss: 0.6112
Epoch 2/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8846 - loss: 2.5007 - val_a
ccuracy: 0.9393 - val_loss: 1.7747
Epoch 3/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8827 - loss: 2.6408 - val_a
ccuracy: 0.9552 - val_loss: 0.2441
Epoch 4/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8931 - loss: 1.6911 - val_a
ccuracy: 0.9781 - val_loss: 0.1503
Epoch 5/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9069 - loss: 1.3572 - val_a
ccuracy: 0.9393 - val_loss: 4.1423
Epoch 6/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9042 - loss: 1.4876 - val_a
ccuracy: 0.9393 - val_loss: 1.3426
Epoch 7/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9139 - loss: 0.9854 - val_a
ccuracy: 0.9660 - val_loss: 0.1898
Epoch 8/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9194 - loss: 0.7925 - val_a
ccuracy: 0.9790 - val_loss: 0.0977
Epoch 9/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9202 - loss: 0.8643 - val_a
ccuracy: 0.9815 - val_loss: 0.0853
Epoch 10/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9339 - loss: 0.5593 - val_a
ccuracy: 0.9842 - val_loss: 0.0812
Epoch 11/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9306 - loss: 0.6468 - val_a
ccuracy: 0.9803 - val_loss: 0.1270
Epoch 12/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9316 - loss: 0.6259 - val_a
ccuracy: 0.7619 - val_loss: 0.4343
Epoch 13/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9443 - loss: 0.4942 - val_a
ccuracy: 0.9393 - val_loss: 0.4455
Epoch 14/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9487 - loss: 0.3548 - val_a
ccuracy: 0.9855 - val_loss: 0.0641
Epoch 15/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9347 - loss: 0.7504 - val_a
ccuracy: 0.9757 - val_loss: 0.1944
Epoch 16/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9600 - loss: 0.2537 - val_a
ccuracy: 0.9393 - val_loss: 0.8264
Epoch 17/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9474 - loss: 0.4776 - val_a
ccuracy: 0.9856 - val_loss: 0.0720
Epoch 18/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9574 - loss: 0.3179 - val_a
ccuracy: 0.9856 - val_loss: 0.0582
Epoch 19/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9650 - loss: 0.1809 - val_a
ccuracy: 0.9856 - val_loss: 0.0575
```

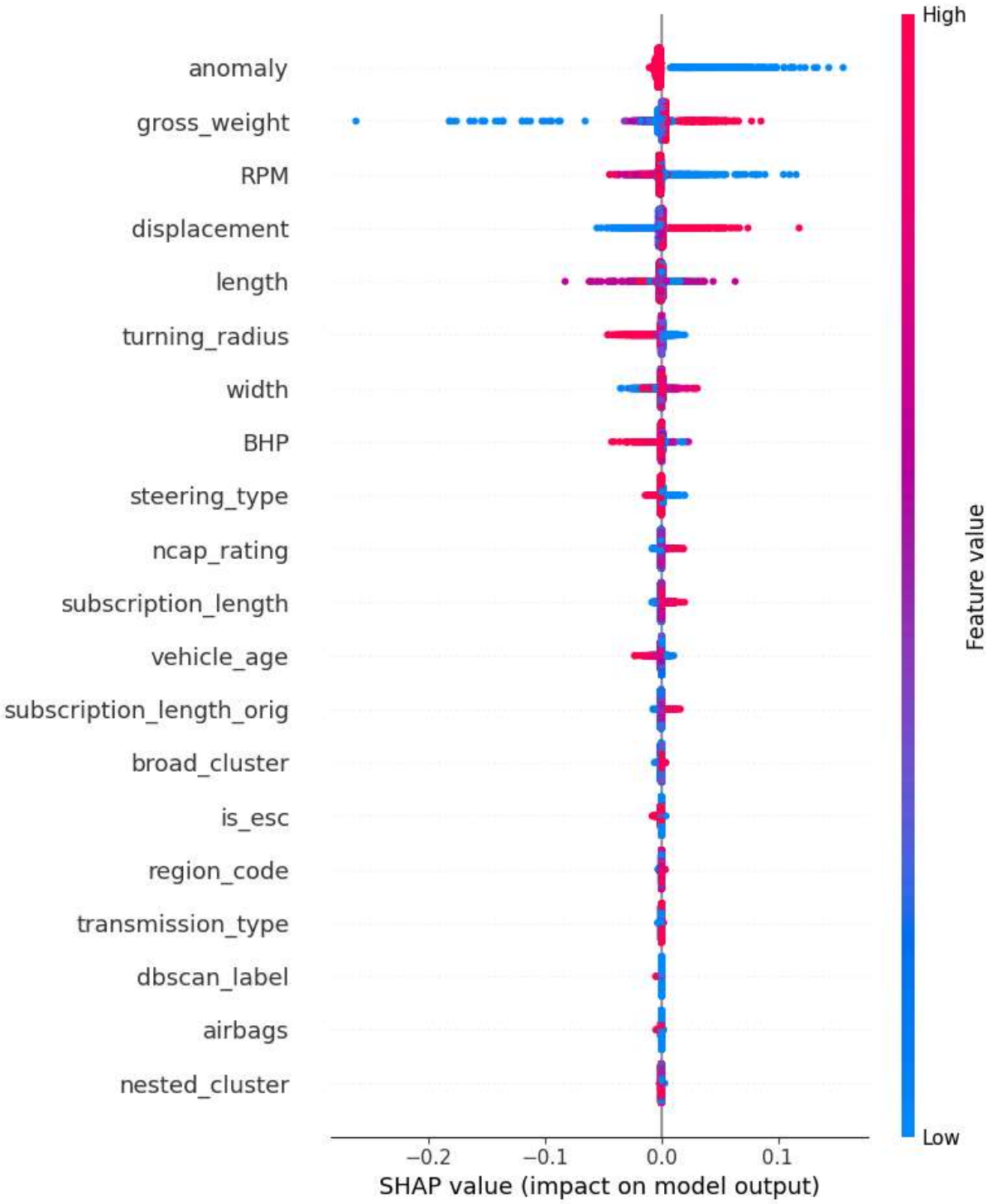
```
Epoch 20/20
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9621 - loss: 0.2250 - val_a
ccuracy: 0.9393 - val_loss: 0.4836
367/367 ━━━━━━━━━━━━━━━━━━━━ 0s 826us/step
[[10973     0]
 [  746     0]]
              precision    recall  f1-score   support

         0.0       0.94      1.00      0.97     10973
         1.0       0.00      0.00      0.00       746

    accuracy                           0.94     11719
   macro avg       0.47      0.50      0.48     11719
weighted avg       0.88      0.94      0.91     11719
```

```
c:\Users\mtbaj\Documents\vscode-workspace\.venv\Lib\site-packages\sklearn\metrics\_c
lassification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\mtbaj\Documents\vscode-workspace\.venv\Lib\site-packages\sklearn\metrics\_c
lassification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\mtbaj\Documents\vscode-workspace\.venv\Lib\site-packages\sklearn\metrics\_c
lassification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [22]:

```
# feature importance analysis
explainer = shap.Explainer(model, x_test)

shap_values = explainer(x_test)

shap.summary_plot(shap_values, x_test)
```

```
PermutationExplainer explainer: 11720it [09:10, 21.03it/s]
```

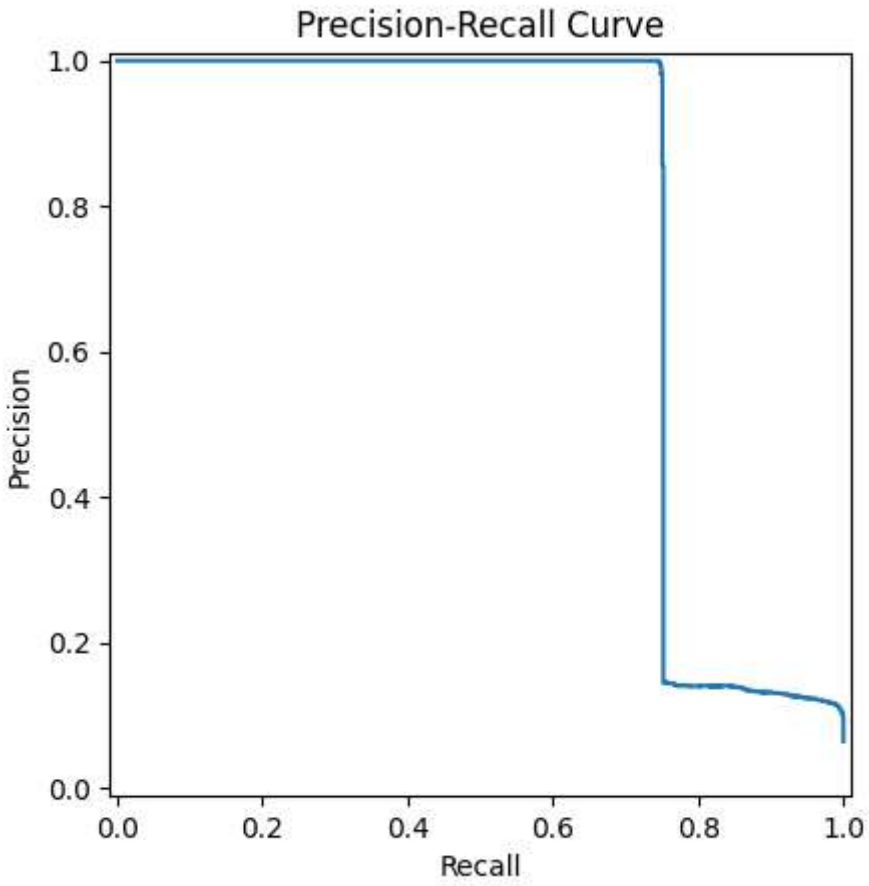


Task 2: Model Evaluation

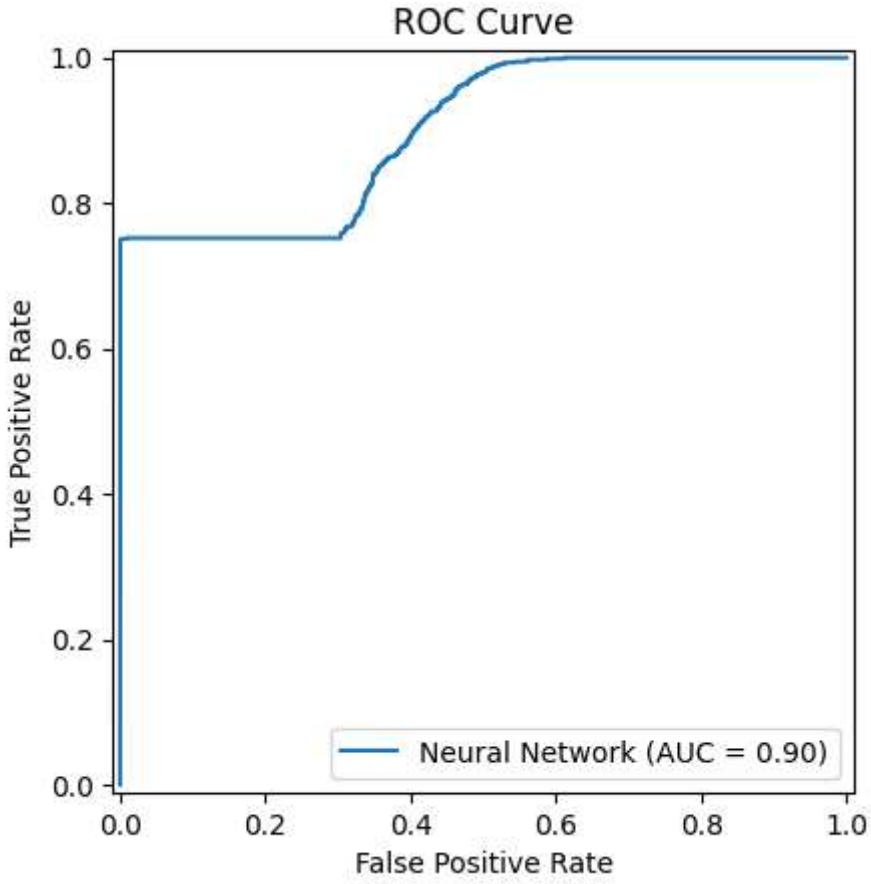
In [23]:
```
# evaluate using:
# precision-recall curves
# Get Predicted Probabilities
y_probs = model.predict(x_test).ravel()
precision, recall, thresholds = precision_recall_curve(y_test, y_probs)

# Compute precision-recall values
disp = PrecisionRecallDisplay(precision=precision, recall=recall)
disp.plot()
```

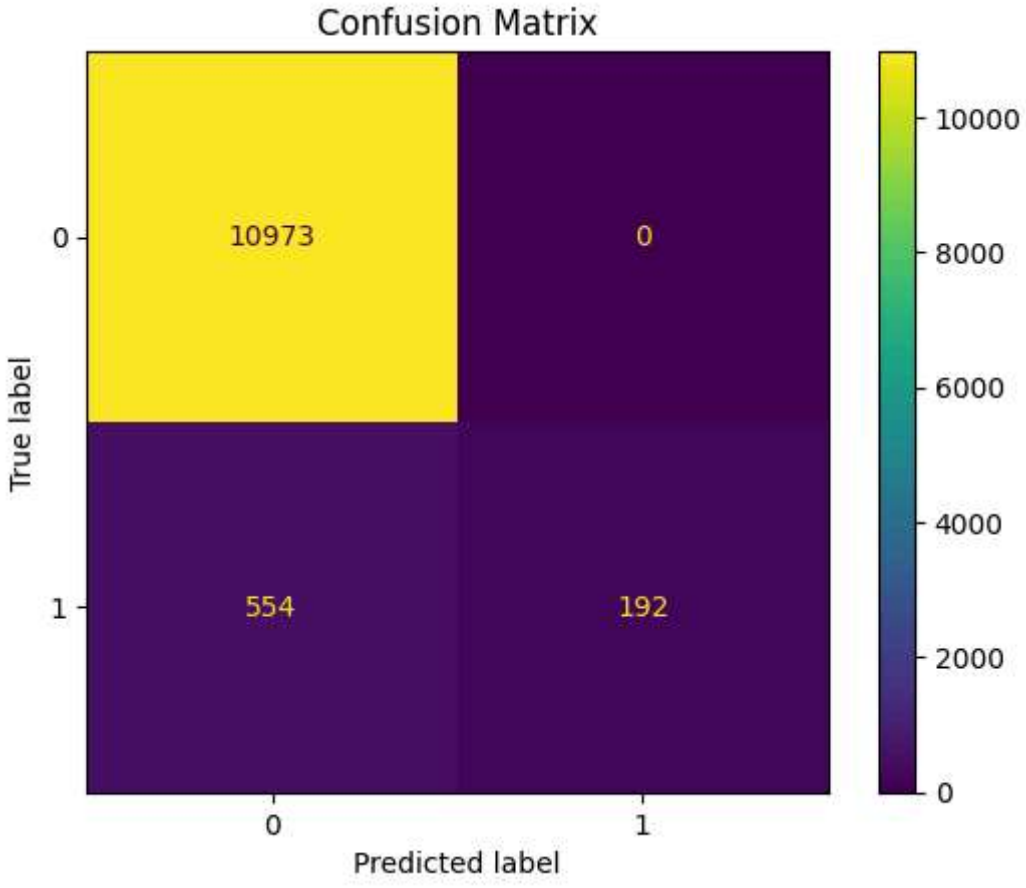
```
plt.title("Precision-Recall Curve")
plt.show()

# ROC curve and AUC score
# Compute ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# Calc AUC Score
roc_auc = auc(fpr, tpr)
print(f"AUC Score: {roc_auc:.2f}")

disp = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc, estimator_name='Neural Ne
disp.plot()
plt.title("ROC Curve")
plt.show()

# confusion matrix
y_pred = (y_probs > 0.05).astype(int)

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0,1])
disp.plot()
plt.title("Confusion Matrix")
plt.show()
```

```
367/367 ━━━━━━━━━━━━━━━━━━━━ 0s 728us/step
```




```
AUC Score: 0.90
```


ROC Curve
True Positive Rate
False Positive Rate
Neural Network (AUC = 0.90)
0.0
0.2
0.4
0.6
0.8
1.0

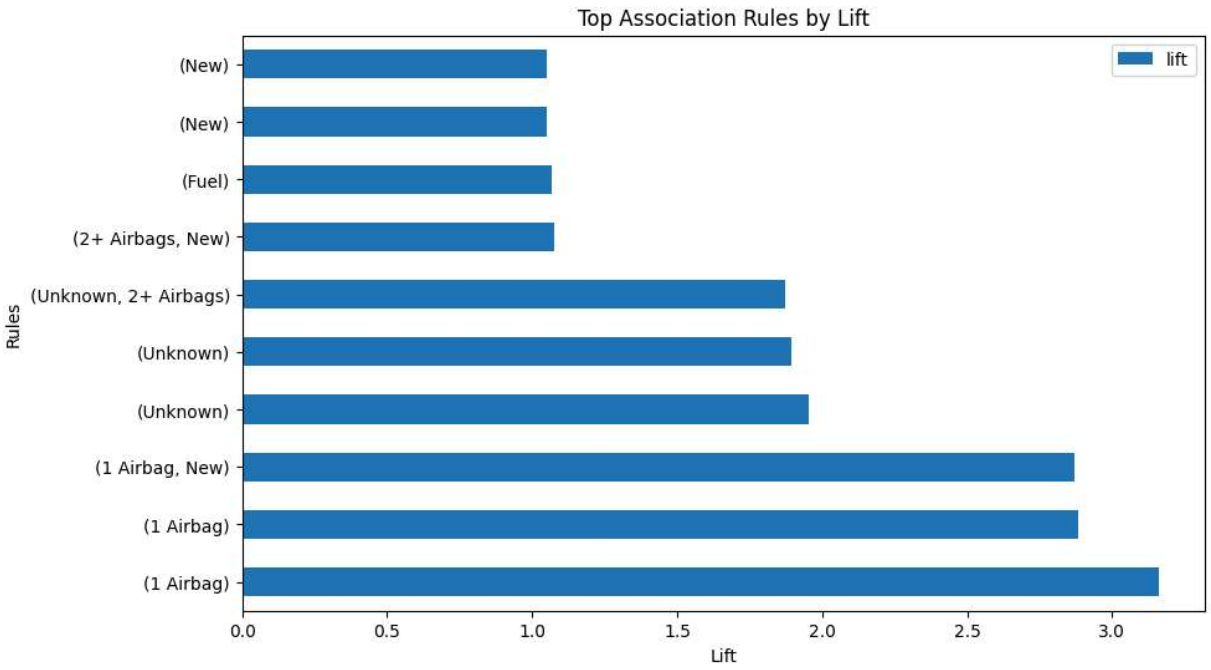

Confusion Matrix
True label
Predicted label
10973
0
554
192
0
1
10000
8000
6000
4000
2000
0

Pattern Modeling

Task 1: Association Rule Mining

In [24]:
```
# mining relationships -- connection between vehicle feat. and claim freq.
selected_features = ['vehicle_age_binned', 'claim_status_binned', 'fuel_type_binned

# Clean data
if "Unknown" not in df['vehicle_age_binned'].cat.categories:
    df['vehicle_age_binned'] = df['vehicle_age_binned'].cat.add_categories(['Unknow
df['claim_status_binned'] = df['claim_status_binned'].fillna('Unknown')
df['vehicle_age_binned'] = df['vehicle_age_binned'].fillna('Unknown')
df['fuel_type_binned'] = df['fuel_type_binned'].fillna('Unknown')
df['airbags_binned'] = df['airbags_binned'].fillna('Unknown')

# Transform into transaction data format
transactions = df[selected_features].apply(lambda row: row.dropna().astype(str).tol
te = TransactionEncoder()
te_data = te.fit(transactions).transform(transactions)
df_transformed = pd.DataFrame(te_data, columns=te.columns_)

item_frequencies = df_transformed.sum(axis=0)
# Check the frequency of specific items
print("Frequency of 'Claim':", item_frequencies.get('Claim', 0))
print("Frequency of 'No_Claim':", item_frequencies.get('No_Claim', 0))

frequent_itemsets = fpgrowth(df_transformed, min_support=0.01, use_colnames=True)
num_itemsets = len(frequent_itemsets)
rules = association_rules(df=frequent_itemsets, num_itemsets=num_itemsets, metric='
#print(rules.head())

filtered_rules = rules[rules['consequents'].apply(lambda x: 'Claim' in x or 'No_Cla
#print(filtered_rules)

filtered_rules.nlargest(10, 'lift').plot(x='antecedents', y='lift', kind='barh', fi
plt.title('Top Association Rules by Lift')
plt.xlabel('Lift')
plt.ylabel('Rules')
plt.show()
```

```
Frequency of 'Claim': 3748
Frequency of 'No_Claim': 54844
```



In [25]:
```
# metrics -- support, confidence, lift
top_rules = filtered_rules.nlargest(10, 'lift')
print("Top 10 Association Rules:")
association_data = pd.DataFrame(top_rules[['antecedents', 'consequents', 'support',
association_data
```

```
Top 10 Association Rules:
```

Out[25]:

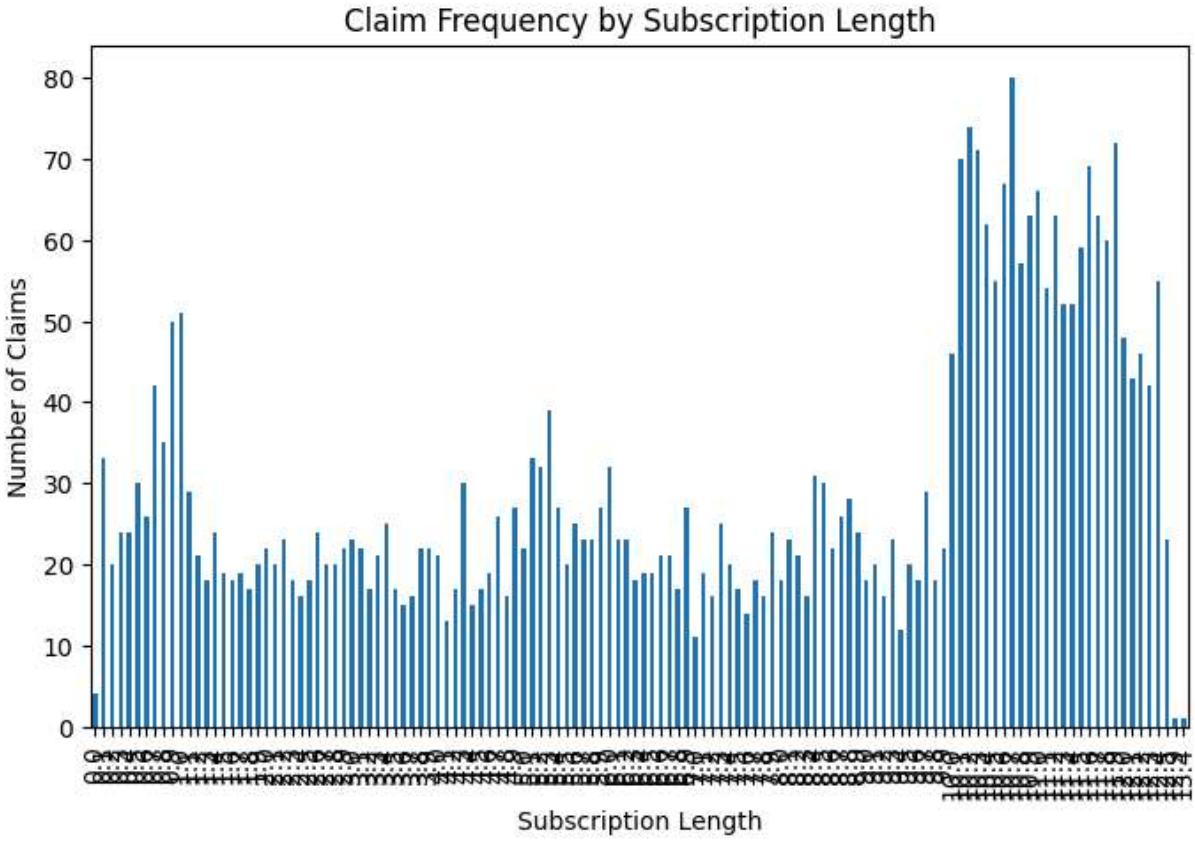
	antecedents	consequents	support	confidence	lift
135	(1 Airbag)	(Other, No_Claim, New)	0.017613	0.853598	3.160443
122	(1 Airbag)	(Other, No_Claim)	0.019388	0.939620	2.883173
134	(1 Airbag, New)	(Other, No_Claim)	0.017613	0.935630	2.870932
111	(Unknown)	(Other, 2+ Airbags, No_Claim)	0.053710	0.598630	1.953057
104	(Unknown)	(Other, No_Claim)	0.055434	0.617843	1.895818
110	(Unknown, 2+ Airbags)	(Other, No_Claim)	0.053710	0.610594	1.873575
44	(2+ Airbags, New)	(Fuel, No_Claim)	0.582076	0.655752	1.074766
46	(Fuel)	(2+ Airbags, No_Claim, New)	0.582076	0.891354	1.068065
35	(New)	(Fuel, No_Claim)	0.582076	0.642134	1.052446
49	(New)	(Fuel, 2+ Airbags, No_Claim)	0.582076	0.642134	1.052446

Task 2: Sequential Pattern Analysis

In [26]:
```
# subscription analysis -- how claim pattern evolves over subscrption_length
sequences = df.sort_values(['policy_id', 'subscription_length_orig']).groupby('poli

sequence_list = sequences.tolist()
```

```
ps = PrefixSpan(sequence_list)
ps.minlen = 2
ps.maxlen = 5

patterns = ps.frequent(2)

print("Frequent Patterns:")
for pattern, support in patterns:
    print(f"Pattern: {pattern}, Support: {support}")

claim_frequency = df[df["claim_status_binned"] == 'Claim'].groupby('subscription_le

claim_frequency.plot(kind='bar', figsize=(8,5), title='Claim Frequency by Subscript
plt.xlabel('Subscription Length')
plt.ylabel('Number of Claims')
plt.show()
```

```
Frequent Patterns:
```



In [27]:
```
# high-risk feat. combinations -- how do feat. combos correlate to higher claims
df['feature_combination'] = df[['vehicle_age_binned', 'airbags_binned', 'region_den
    lambda row: f"{row['vehicle_age_binned']}, {row['airbags_binned']}, {row['regio
)

# Calculate claim frequency for each combination
combination_stats = df.groupby('feature_combination')['claim_status_binned'].value_

# Calculate claim rate
combination_stats['Claim_Rate'] = combination_stats['Claim'] / (combination_stats['
```

```
# Sort by claim rate
high_risk_combinations = combination_stats.sort_values(by='Claim_Rate', ascending=F
print(high_risk_combinations)

# Plot high-risk combinations
top_combinations = high_risk_combinations.head(5)

top_combinations['Claim_Rate'].plot(kind='bar', figsize=(8, 5), title='High-Risk Fe
plt.xlabel('Feature Combination')
plt.ylabel('Claim Rate')
plt.xticks(rotation=45, ha='right')
plt.show()
```

claim_status_binned	Claim	No_Claim	Claim_Rate
feature_combination			
Unknown, 2+ Airbags, High	557	4597	0.108071
New, 1 Airbag, High	71	1032	0.064370
New, 2+ Airbags, High	3111	48898	0.059817
Middle-aged, 2+ Airbags, High	7	208	0.032558
Unknown, 1 Airbag, High	2	101	0.019417
Middle-aged, 1 Airbag, High	0	2	0.000000
Old, 2+ Airbags, High	0	5	0.000000
Old, 1 Airbag, High	0	1	0.000000