DT503_3_Traffic_Accident_Data_Analysis_Walkthrough

November 12, 2024

1 Traffic Accident Data Analysis

This notebook explores and analyses a subset of the 'dft-road-casualty-statistics-collision-2023' traffic accident dataset, filtered to a specific region. Using techniques such as Principal Component Analysis (PCA), Factor Analysis, Cluster Analysis and Discriminant Analysis. The goal is to uncover patterns, reduce data dimensionality and develop a reliable classification model.

1.1 Analysis Techniques Covered:

- **Principal Component Analysis (PCA)**: for dimensionality reduction
- **Factor Analysis**: to identify latent factors
- **Cluster Analysis**: to find natural groupings in the data
- **Discriminant Analysis**: to develop a classification model

1.2 Step 1: Load and Inspect Data

Let's begin by loading the dataset and examining its structure.

[2]:
```python
import pandas as pd

# Load the dataset
data = pd.
 ↪read_csv('dft-road-casualty-statistics-collision-2023_Devon_and_Cornwall.
 ↪csv')

# Display the first few rows and basic info
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2806 entries, 0 to 2805
Data columns (total 37 columns):
 #   Column                                       Non-Null Count  Dtype
---  ------                                       --------------  -----
 0   accident_index                               2806 non-null   float64
 1   accident_year                                2806 non-null   int64
 2   accident_reference                           2806 non-null   int64
 3   location_easting_osgr                        2806 non-null   int64
 4   location_northing_osgr                       2806 non-null   int64
 5   longitude                                    2806 non-null   float64
 6   latitude                                     2806 non-null   float64
 7   police_force                                 2806 non-null   int64
 8   accident_severity                            2806 non-null   int64
 9   number_of_vehicles                           2806 non-null   int64
 10  number_of_casualties                         2806 non-null   int64
 11  date                                         2806 non-null   object
 12  day_of_week                                  2806 non-null   int64
 13  time                                         2806 non-null   object
 14  local_authority_district                     2806 non-null   int64
 15  local_authority_ons_district                 2806 non-null   object
 16  local_authority_highway                      2806 non-null   object
 17  first_road_class                             2806 non-null   int64
 18  first_road_number                            2806 non-null   int64
 19  road_type                                    2806 non-null   int64
 20  speed_limit                                  2806 non-null   int64
 21  junction_detail                              2806 non-null   int64
 22  junction_control                             2806 non-null   int64
 23  second_road_class                            2806 non-null   int64
 24  second_road_number                           2806 non-null   int64
 25  pedestrian_crossing_human_control            2806 non-null   int64
 26  pedestrian_crossing_physical_facilities      2806 non-null   int64
 27  light_conditions                             2806 non-null   int64
 28  weather_conditions                           2806 non-null   int64
 29  road_surface_conditions                      2806 non-null   int64
 30  special_conditions_at_site                   2806 non-null   int64
 31  carriageway_hazards                          2806 non-null   int64
 32  urban_or_rural_area                          2806 non-null   int64
 33  did_police_officer_attend_scene_of_accident  2806 non-null   int64
 34  trunk_road_flag                              2806 non-null   int64
 35  lsoa_of_accident_location                    2806 non-null   object
 36  enhanced_severity_collision                  2806 non-null   int64
dtypes: float64(3), int64(29), object(5)
memory usage: 811.2+ KB
```

```
[2]:    accident_index  accident_year  accident_reference  location_easting_osgr  \
     0    2.023500e+12           2023           501258810                 251899
     1    2.023500e+12           2023           501259203                 285979
     2    2.023500e+12           2023           501259228                 202261
     3    2.023500e+12           2023           501259501                 251599
     4    2.023500e+12           2023           501259642                 244791

        location_northing_osgr  longitude   latitude  police_force  \
     0                   51705  -4.082892  50.346591            50
     1                   64004  -3.607870  50.464747            50
     2                   52632  -4.780271  50.340338            50
     3                   59514  -4.090189  50.416697            50
```

```
4                   57992  -4.185314  50.401258            50

   accident_severity  number_of_vehicles  …  light_conditions  \
0                  2                   2  …                 1
1                  3                   2  …                 1
2                  3                   2  …                 1
3                  3                   2  …                 4
4                  3                   2  …                 1

  weather_conditions  road_surface_conditions special_conditions_at_site  \
0                  1                        2                          0
1                  4                        2                          7
2                  5                        2                          0
3                  5                        2                          0
4                  2                        2                          0

   carriageway_hazards urban_or_rural_area  \
0                    0                   2
1                    0                   2
2                    0                   1
3                    0                   1
4                    0                   1

  did_police_officer_attend_scene_of_accident  trunk_road_flag  \
0                                           1                2
1                                           1                2
2                                           2                2
3                                           3                2
4                                           1                2

   lsoa_of_accident_location  enhanced_severity_collision
0                  E01015129                            6
1                  E01020170                            3
2                  E01019041                            3
3                  E01015090                            3
4                  E01015143                            3

[5 rows x 37 columns]
```

1.3 Step 2: Principal Component Analysis (PCA)

PCA reduces the data's dimensionality by finding components that explain most of the variance. This helps simplify complex datasets.

[3]:
```python
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

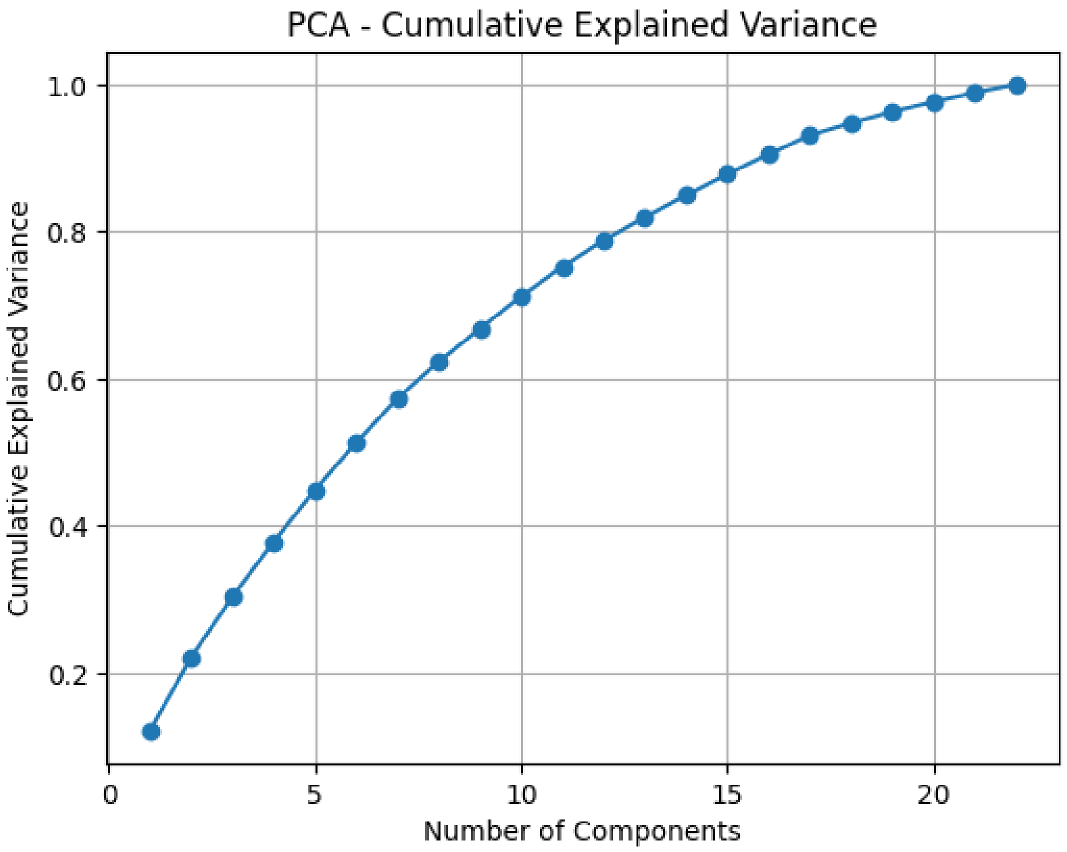
```
# Selecting relevant columns for PCA
numerical_columns = [
    'location_easting_osgr', 'location_northing_osgr', 'accident_severity',
    'number_of_vehicles', 'number_of_casualties', 'day_of_week',
    'first_road_class', 'road_type', 'speed_limit', 'junction_detail',
    'junction_control', 'pedestrian_crossing_human_control',
    'pedestrian_crossing_physical_facilities', 'light_conditions',
    'weather_conditions', 'road_surface_conditions',␣
  ↪'special_conditions_at_site',
    'carriageway_hazards', 'urban_or_rural_area',
    'did_police_officer_attend_scene_of_accident', 'trunk_road_flag',
    'enhanced_severity_collision'
]

# Standardising the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data[numerical_columns])

# Applying PCA
pca = PCA()
pca.fit(data_scaled)
explained_variance_ratio = pca.explained_variance_ratio_
explained_variance_ratio
```

```
[3]: array([0.12133061, 0.10018245, 0.08198755, 0.07498831, 0.07042889,
            0.0641444 , 0.06038357, 0.04953602, 0.04505504, 0.04405497,
            0.04015928, 0.03555746, 0.0319206 , 0.02973638, 0.02878374,
            0.02691546, 0.02580875, 0.01677796, 0.01536993, 0.01344648,
            0.01241099, 0.01102115])
```

1.3.1 PCA Explained Variance

We can visualise the cumulative variance explained by each component.

```
[4]: import matplotlib.pyplot as plt

# Plotting cumulative explained variance
plt.plot(range(1, len(explained_variance_ratio) + 1), pca.
  ↪explained_variance_ratio_.cumsum(), marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('PCA - Cumulative Explained Variance')
plt.grid()
plt.show()
```



The cumulative explained variance plot shows the proportion of variance captured by each component.

1.3.2 Interpretation

The original dataset contained 22 numerical components relevant to accident conditions (after selecting and standardising them).

PCA revealed that the first 5 components explained approximately 45% of the total variance in the data. This means that instead of using all 22 original variables, we can focus on these 5 main components for a simpler yet representative view of the dataset.

The reduction from 22 to 5 components enables us to capture the primary patterns without the full complexity of the original data, facilitating clearer analysis in the subsequent steps.

1.4 Step 3: Factor Analysis

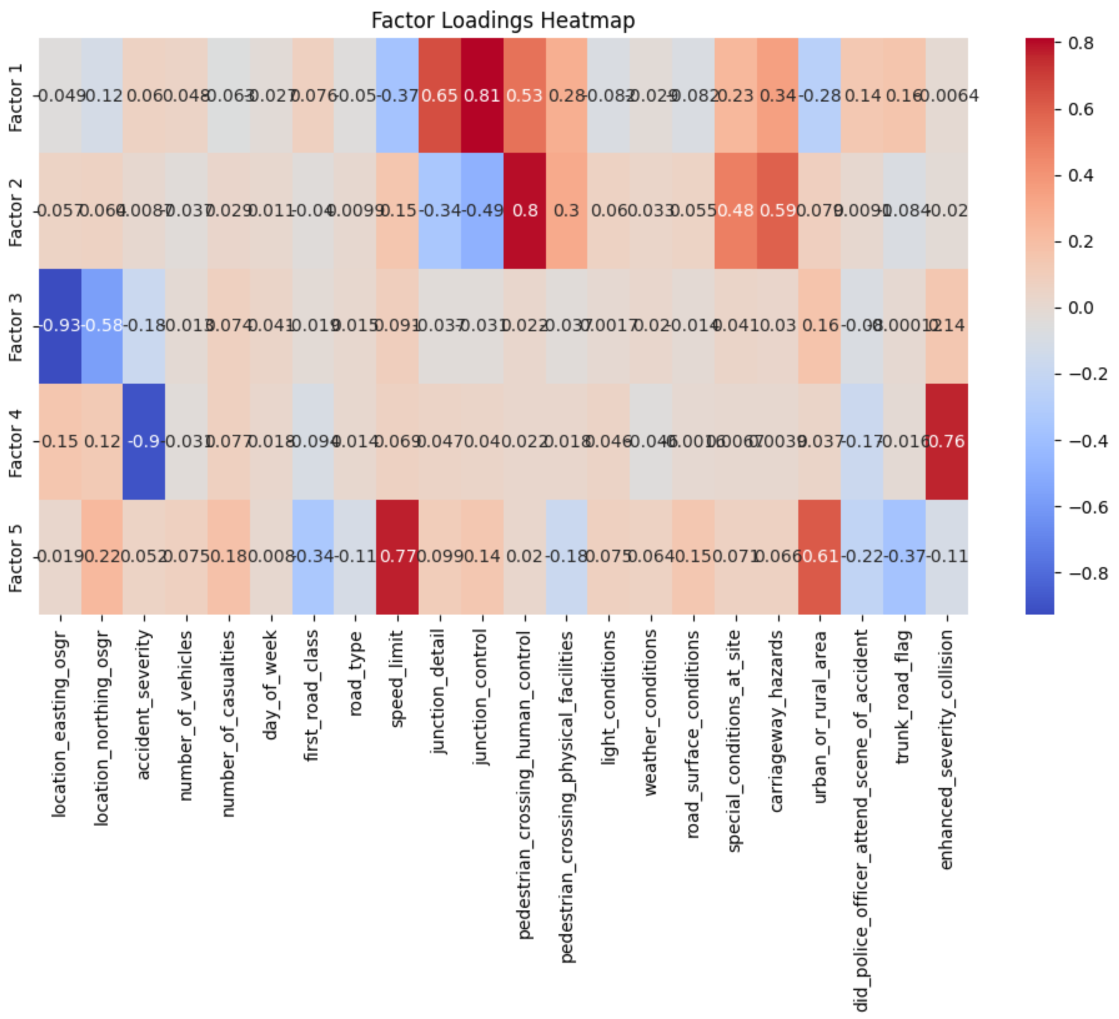
Factor Analysis identifies latent factors within the data. These factors help explain the underlying structure of relationships among variables.

1.4.1 Factor Loadings Heatmap

The heatmap below shows the factor loadings, highlighting which variables contribute most to each factor.

[5]:
```python
from sklearn.decomposition import FactorAnalysis

# Applying Factor Analysis
factor_analysis = FactorAnalysis(n_components=5, random_state=0)
factors = factor_analysis.fit_transform(data_scaled)

# Analyzing factor loadings
factor_loadings = factor_analysis.components_
```

[6]:
```python
import seaborn as sns

# Creating a heatmap of factor loadings
plt.figure(figsize=(12, 6))
sns.heatmap(factor_loadings, annot=True, cmap='coolwarm', yticklabels=[f'Factor␣
 ↪{i+1}' for i in range(factor_loadings.shape[0])],␣
 ↪xticklabels=numerical_columns)
plt.title('Factor Loadings Heatmap')
plt.show()
```



The heatmap of factor loadings shows the strength of association between each variable and the latent factors identified.

- Each factor (e.g., Factor 1, Factor 2) has strong loadings for certain variables, suggesting specific accident patterns.
- One factor may have high loadings on variables related to road type and junction details, suggesting an underlying pattern linked to road infrastructure.

1.4.2 Interpretation

Factor analysis helps us to identify groups of related variables, enabling a deeper understanding of accident characteristics. Latent factors simplify the dataset by grouping variables with shared patterns, enhancing data interpretation.

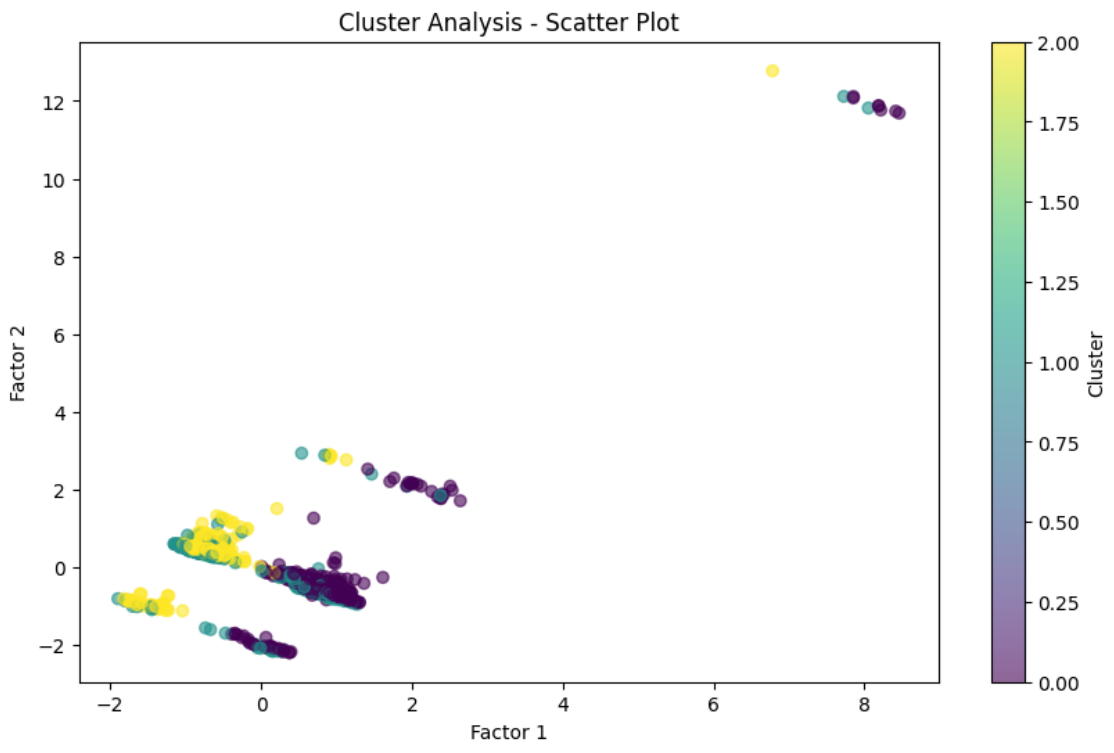
1.5 Step 4: Cluster Analysis

Using the factors from the Factor Analysis step, we can apply clustering techniques to segment the data into distinct groups, which may highlight different types of accident profiles.

1.5.1 Cluster Scatter Plot

The scatter plot below shows the clusters on the first two principal components to help visualise how the data is grouped.

[7]:
```
from sklearn.cluster import KMeans

# Performing KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=0)
clusters = kmeans.fit_predict(factors)

# Adding the cluster labels to the original data
data['Cluster'] = clusters

# Scatter plot of clusters
plt.figure(figsize=(10, 6))
plt.scatter(factors[:, 0], factors[:, 1], c=clusters, cmap='viridis',␣
  ↪marker='o', alpha=0.6)
plt.xlabel('Factor 1')
plt.ylabel('Factor 2')
plt.title('Cluster Analysis - Scatter Plot')
plt.colorbar(label='Cluster')
plt.show()
```



The scatter plot shows clusters of data points based on the first two factors.

- Data points are grouped into three clusters, each representing distinct patterns within the accident data.
- Clusters indicate that certain factors, such as location and severity, may distinguish types of accidents, possibly related to urban vs. rural settings or severity levels.

1.5.2 Interpretation

Clustering reveals natural groupings, which could help identify accident types with similar profiles. These insights could be useful for targeted safety interventions based on the specific characteristics of each cluster.

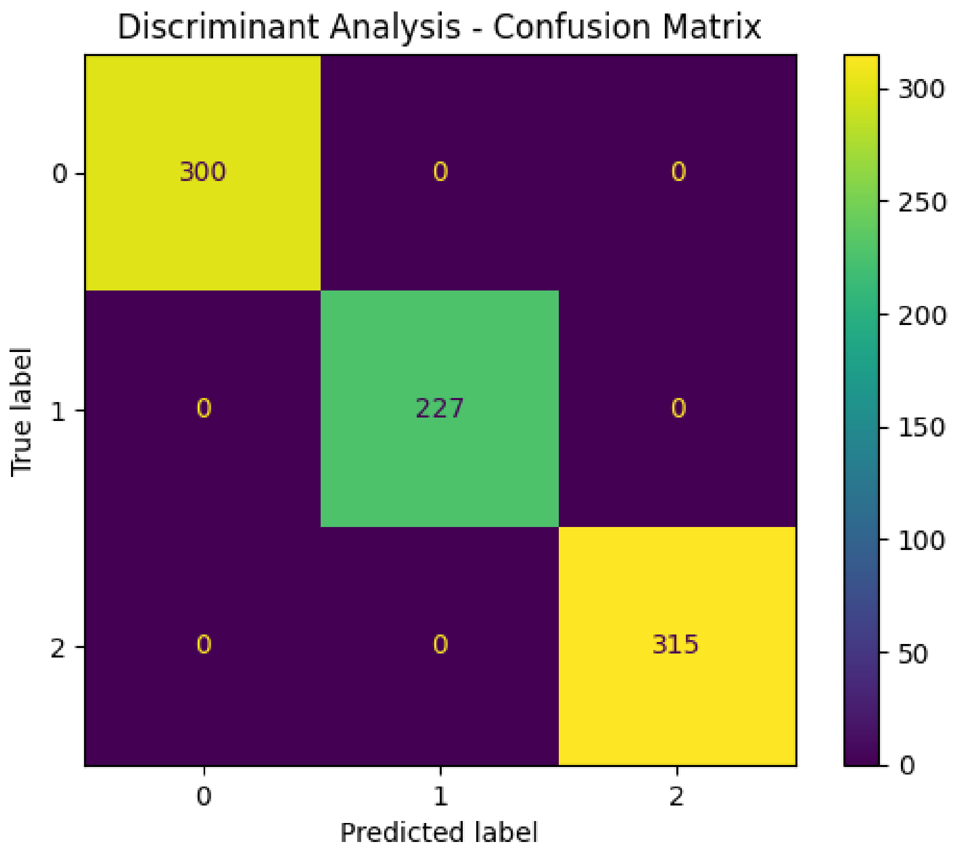
1.6 Step 5: Discriminant Analysis

Using the clusters as the target variable, we apply Discriminant Analysis to develop a classification model. This model can help predict the cluster for new data points.

1.6.1 Confusion Matrix

The confusion matrix below shows the accuracy of the Discriminant Analysis model in predicting cluster membership.

[8]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, ConfusionMatrixDisplay

# Preparing data
X = factors
y = clusters

# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=0)

# Applying LDA
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
y_pred = lda.predict(X_test)

# Displaying confusion matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('Discriminant Analysis - Confusion Matrix')
plt.show()

# Classification report
print(classification_report(y_test, y_pred))
```



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       300
           1       1.00      1.00      1.00       227
           2       1.00      1.00      1.00       315

    accuracy                           1.00       842
   macro avg       1.00      1.00      1.00       842
weighted avg       1.00      1.00      1.00       842
```

The confusion matrix shows the model's accuracy in classifying data points into clusters.

- The model achieved 100% accuracy, with no misclassifications among clusters, indicating strong separation between the clusters.
- The classification report shows perfect precision, recall and F1 scores, reinforcing the model's reliability.

This high accuracy suggests that the clusters identified in Step 4 are well-defined, with distinct characteristics that the model can reliably predict. This classification model could potentially be applied to new data points to predict accident types based on the patterns observed.

1.7 Data Export for Power BI Visualisation

To enhance our data analysis in Power BI, we will now export the following:

- Original data with added PCA components (reduced dimensions): This enables pattern visualisation without all original variables.
- Cluster labels: Include labels assigned to each data point by the K-means model; this allows visualisation of each accident record's cluster category, aiding in profiling accident types.
- Factor scores (from factor analysis): Export the dataset with factor scores to visualise latent patterns identified by factor analysis.
- Classification labels (if using discriminant analysis): If discriminant analysis was used to classify clusters, add these predicted labels; this allows Power BI to categorise and filter new data points by predicted cluster.

These exports will provide Power BI with the necessary data to create insightful visualisations and analyses of accident patterns and classifications.

[9]:
```python
# Add PCA components (first 5 for example), clusters, and factors to the␣
 ↪original data
data_export = data.copy()
data_export[['PC1', 'PC2', 'PC3', 'PC4', 'PC5']] = pca.transform(data_scaled)[:
 ↪, :5]  # First 5 components
data_export['Cluster'] = clusters  # Clustering labels
data_export[['Factor1', 'Factor2', 'Factor3', 'Factor4', 'Factor5']] = factors[:
 ↪, :5]  # First 5 factor scores
```

```python
# Generate classification labels based on factor scores
classification_labels = lda.predict(factors)

# Add classification labels to the DataFrame
data_export['Classification_Label'] = classification_labels

# Save to CSV for Power BI
data_export.to_csv('accident_analysis_export.csv', index=False)
```