In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("last_two_years_accidents.csv")
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2009085 entries, 0 to 2009084
Data columns (total 41 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   ID                     object 
 1   Source                 object 
 2   Severity               int64  
 3   Start_Time             object 
 4   End_Time               object 
 5   Start_Lat              float64
 6   Start_Lng              float64
 7   Distance(mi)           float64
 8   Street                 object 
 9   City                   object 
 10  County                 object 
 11  State                  object 
 12  Zipcode                object 
 13  Country                object 
 14  Timezone               object 
 15  Airport_Code           object 
 16  Weather_Timestamp      object 
 17  Temperature(F)         float64
 18  Humidity(%)            float64
 19  Pressure(in)           float64
 20  Visibility(mi)         float64
 21  Wind_Direction         object 
 22  Wind_Speed(mph)        float64
 23  Weather_Condition      object 
 24  Amenity                bool   
 25  Bump                   bool   
 26  Crossing               bool   
 27  Give_Way               bool   
 28  Junction               bool   
 29  No_Exit                bool   
 30  Railway                bool   
 31  Roundabout             bool   
 32  Station                bool   
 33  Stop                   bool   
 34  Traffic_Calming        bool   
 35  Traffic_Signal         bool   
 36  Turning_Loop           bool   
 37  Sunrise_Sunset         object 
 38  Civil_Twilight         object 
 39  Nautical_Twilight      object 
 40  Astronomical_Twilight  object 
dtypes: bool(13), float64(8), int64(1), object(19)
memory usage: 454.1+ MB
```

In [4]:
```
severity_labels = {
    1: 'Not Severe',
    2: 'Not Severe',
    3: 'Severe',
    4: 'Severe'
}
```

In [5]:
```
df['Severity'] = df['Severity'].map(severity_labels)
```

In [6]:
```
size = df['Severity'].value_counts()['Severe']
```

In [7]:
```
df_balanced_severity = pd.DataFrame()
```

In [8]:
```
df_balanced_severity = df.groupby('Severity', group_keys = False).apply(lambda x: x.sample(size, random_state = 30))
```

In [9]:
```
df_balanced_severity['Severity'].value_counts()
```

Out[9]:
```
Severity
Not Severe    128292
Severe        128292
Name: count, dtype: int64
```

In [10]:
```
columns_to_keep = ['Temperature(F)', 'Humidity(%)', 'Pressure(in)',
           'Visibility(mi)','Severity']

new_df = df_balanced_severity[columns_to_keep]
```

In [11]:
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 256584 entries, 461355 to 109670
Data columns (total 5 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Temperature(F)  256584 non-null  float64
 1   Humidity(%)     256584 non-null  float64
 2   Pressure(in)    256584 non-null  float64
 3   Visibility(mi)  256584 non-null  float64
 4   Severity        256584 non-null  object 
dtypes: float64(4), object(1)
memory usage: 11.7+ MB
```

In [12]:
```
256584 /2
```

Out[12]:
```
128292.0
```

In [13]:
```
new_df.head()
```

Out[13]:

	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Severity
461355	55.0	55.0	29.06	10.0	Not Severe
1340271	93.0	56.0	29.98	10.0	Not Severe
1291327	70.0	93.0	29.05	10.0	Not Severe
170756	63.0	27.0	29.03	10.0	Not Severe
1675489	41.0	100.0	30.28	10.0	Not Severe

In [14]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

new_df['Severity'] = label_encoder.fit_transform(new_df['Severity'])
print(new_df)
```

```
        Temperature(F)  Humidity(%)  Pressure(in)  Visibility(mi)  Severity
461355        55.000000    55.000000     29.060000       10.000000         0
1340271       93.000000    56.000000     29.980000       10.000000         0
1291327       70.000000    93.000000     29.050000       10.000000         0
170756        63.000000    27.000000     29.030000       10.000000         0
1675489       41.000000   100.000000     30.280000       10.000000         0
...                 ...          ...           ...             ...       ...
935219        61.000000    89.000000     28.880000       10.000000         1
1363937       74.000000    85.000000     29.020000       10.000000         1
139521        55.000000    83.000000     30.210000       10.000000         1
207978        61.000000    22.000000     29.570000       10.000000         1
109670        61.663286    64.831041     29.538986        9.090376         1

[256584 rows x 5 columns]
C:\Users\trina\AppData\Local\Temp\ipykernel_155624\1983017577.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Severity'] = label_encoder.fit_transform(new_df['Severity'])
```

In [15]:
```
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

In [16]:
```
sample_1 = new_df.sample(10000, random_state=42)
```

In [17]:
```
y_sample1 = sample_1["Severity"]
x_sample1 = sample_1.drop("Severity", axis=1)
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(x_sample1, y_sample1, random_state=42)
```

In [19]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.utils import shuffle
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout
```

In [20]:
```
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))  # Assuming 'Severity' is binary
```

```
C:\Users\trina\anaconda3\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)
```

In [22]:
```
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
```

```
optimizer = Adam(learning_rate=0.0001)
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

In [23]:
```
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
```

In [24]:
```
history = model.fit(X_train, y_train, epochs=5, batch_size=64, verbose=1, validation_split=0.2, callbacks=[early_stoppi
```

```
Epoch 1/5
94/94 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.4928 - loss: 4.0070 - val_accuracy: 0.5133 - val_loss: 0.8929
Epoch 2/5
94/94 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4992 - loss: 2.7520 - val_accuracy: 0.5007 - val_loss: 0.8000
Epoch 3/5
94/94 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4962 - loss: 1.7773 - val_accuracy: 0.4813 - val_loss: 0.7583
Epoch 4/5
94/94 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5085 - loss: 1.1584 - val_accuracy: 0.5173 - val_loss: 0.7285
Epoch 5/5
94/94 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5110 - loss: 0.9144 - val_accuracy: 0.5173 - val_loss: 0.7093
```

In [25]:
```
loss, accuracy = model.evaluate(X_test, y_test, verbose=1)
print("Accuracy:", accuracy)
```

```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.5000 - loss: 0.7059
Accuracy: 0.5139999985694885
```

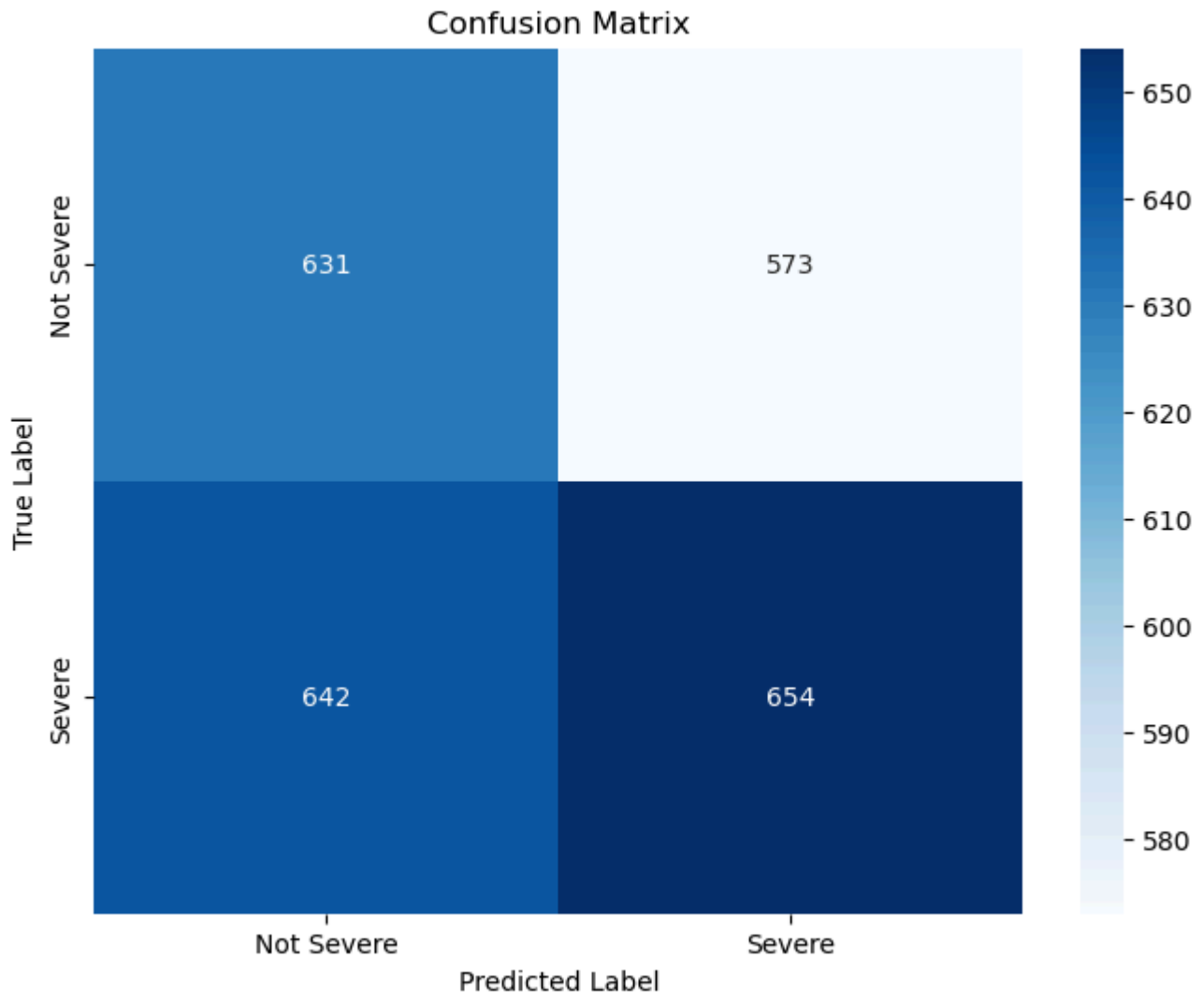
In [26]:
```
y_pred_proba = model.predict(X_test)
y_pred = (y_pred_proba > 0.5).astype(int)
```

```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

In [27]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[631 573]
 [642 654]]
```

In [28]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g',
            xticklabels=['Not Severe', 'Severe'],
            yticklabels=['Not Severe', 'Severe'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



In [29]:
```
X_train
```

Out[29]:

	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)
11047	73.000000	87.000000	29.960000	10.000000
160100	52.000000	34.000000	29.640000	10.000000
43148	83.000000	58.000000	29.420000	10.000000
124660	27.000000	89.000000	29.870000	10.000000
1563968	71.000000	51.000000	29.460000	9.090376
...	...	...	...	...
1112364	63.000000	77.000000	28.530000	10.000000
710995	46.000000	57.000000	29.700000	10.000000
179146	49.000000	100.000000	26.250000	0.000000
291023	61.663286	64.831041	29.538986	9.090376
5582	71.000000	53.000000	29.400000	10.000000

7500 rows × 4 columns

In [30]:
```
X_test
```

Out[30]:

	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)
877746	48.0	96.0	29.79	10.0
536569	50.0	59.0	28.51	10.0
1368515	36.0	67.0	26.08	10.0
403407	42.0	51.0	29.13	10.0
1984804	24.0	57.0	29.57	10.0
...	...	...	...	...
1911932	81.0	62.0	30.14	10.0
767912	68.0	87.0	29.50	10.0
220349	59.0	42.0	29.26	10.0
701700	43.0	89.0	30.29	10.0
1308101	79.0	77.0	29.79	10.0

2500 rows × 4 columns

In [31]:
```
y_train
```

Out[31]:
```
11047      1
160100     1
43148      0
124660     1
1563968    1
          ..
1112364    0
710995     0
179146     1
291023     0
5582       1
Name: Severity, Length: 7500, dtype: int32
```

In [32]:
```
y_test
```

Out[32]:
```
877746     0
536569     1
1368515    0
403407     0
1984804    0
          ..
1911932    0
767912     1
220349     1
701700     0
1308101    0
Name: Severity, Length: 2500, dtype: int32
```

In [ ]: