92

Selecting columns and creating a copy to avoid modifying the original DF

grouping the dataframe by road name

Resets the index of the resulting Series and converts it into a df with a new column named  
  
those actions resulting a new 2 columns df, where ‘incident count’ column represent the number of incidents per road.  
  
merging the columns to the selected columns on ‘road\_name’ to combine the data of the new column (‘incident count’)

93

Label encoder converts the categorical ‘road\_name’ column into numerical labels suitable for machine learning algorithms

94

X represents the selected columns to use an input: 'road\_name', 'weather\_code', 'risk\_level\_wmo', and 'incident\_count'.

Y is the target variable, where 'risk\_category\_encoded' is used as the target variable that model will try to predict based on the input features.

95

Divide the dataset into training and testing sets, ensuring that the model can be trained on one part of the data and evaluated on another part to test its performance (20%).

96-98

Plotting class distribution before and after SMOTE and using the SMOTE method to, ensure balanced representation of the classes in the target variable column ('risk\_category\_encoded').

99

Creating and training Random Forest Classifier because of its ability to handle various of data type, balanced and imbalanced data sets. In addition, Random Forest offers benefits such as reducing overfitting, handling missing values, and providing feature importance, making it a suitable choice for classification task.

100

Making predictions on the test data and evaluate the predictions by comparing the true labels (y\_test) with the predicted labels by the machine learning algorithm (y\_pred\_test\_rf).

Generating classification report that includes precision, recall, f1-score, and support for each class.

Precision: The ratio of correctly predicted positive observations to the total predicted positives.

Recall: The ratio of correctly predicted positive observations to all observations in the actual class.

F1-Score: The weighted average of precision and recall. It considers both false positives and false negatives.

Support: The number of actual occurrences of the class in the dataset.

40.37% of the predictions made by the model are correct. This low accuracy indicates that the model may not be performing well on this classification task.

101

Plotting confusion matrix and examining the error analysis point out a large proportion of Class 0 and Class 2 instances are being confused with each other and with Class 1. This suggests that the features used aren’t capturing the distinctions between these classes. Data preprocessing and better features selecting will provide effective and reliable machine learning model.

**Iteration 2**

Trying different method to achieve higher accuracy by adding additional preprocessing step, weights, and fine tuning for the model. In iteration 2, we count the total number of accidents per road, assign weights to each incident type according to its severity and use the machine learning model to classify risk level according to the new selected features. By calculating the total number of incidents per road, we can identify roads with high incident frequencies and potentially investigate the causes or implement measures to reduce incidents.

109

This line counts the number of occurrences of each unique value in the road\_name column. It returns 2 columns where the index is the unique road names and the values are the counts of incidents.

incident\_counts **=** df['road\_name']**.**value\_counts()**.**reset\_index()

incident\_counts**.**columns **=** ['road\_name', 'total\_incidents']

By calculating the total number of incidents per road, you can identify roads with high incident frequencies and potentially investigate the causes or implement measures to reduce incidents.

incident\_types\_per\_road **=** df**.**groupby(['road\_name', 'incident\_severity'])**.**size()**.**unstack(fill\_value**=**0)**.**reset\_index()

create a data frame that shows the counts of each type of incident severity for each road.

road\_incident\_data **=** pd**.**merge(incident\_counts, incident\_types\_per\_road, on**=**'road\_name')

This code merges two data frames, incident\_counts and incident\_types\_per\_road, based on the road\_name column.

severity\_weights **=** {

'HA1': 1, 'HB1': 1, 'HC1': 1, 'HC2': 1, 'HC3': 1, 'HC4': 1, 'HC5': 1, 'SP1': 1,

'HA2': 2, 'HA3': 2, 'HB2': 2, 'HB3': 2, 'HC6': 2, 'HC7': 2, 'HC8': 2, 'HC10': 2, 'HC13': 2, 'HC14': 2, 'HC11': 2, 'SP2': 2, 'SP3': 2,

'HC15': 3, 'HC16': 3, 'HC17': 3, 'HC18': 3, 'HC19': 3, 'HC20': 3, 'HC21': 3, 'SP4': 3, 'SP5': 3

}

Defines a dictionary called severity\_weights which assigns a weight to each type of incident severity. The weights indicate the severity of each type of incident, with higher weights representing more severe incidents.

**for** severity, weight **in** severity\_weights**.**items():

**if** severity **in** road\_incident\_data**.**columns:

road\_incident\_data[f"{severity}\_weighted"] **=** road\_incident\_data[severity] **\*** weight

iterates through the severity\_weights dictionary and applies the weights to the corresponding columns in the road\_incident\_data data frame to calculate the weighted counts for each incident severity type.

incident\_columns **=** [col **for** col **in** road\_incident\_data**.**columns **if** 'weighted' **in** col]

road\_incident\_data['severity\_score'] **=** road\_incident\_data[incident\_columns]**.**sum(axis**=**1)

This code snippet selects the columns that contain weighted incident counts and calculates a severity\_score for each row in the road\_incident\_data data frame by summing these weighted counts.

bins **=** [**-**float('inf'), 7000, 34000, float('inf')]

labels **=** ['low', 'mid', 'high']

road\_incident\_data['risk\_level'] **=** pd**.**cut(road\_incident\_data['severity\_score'], bins**=**bins, labels**=**labels)

This code snippet categorizes the severity\_score for each road into risk levels (low, mid, high) based on predefined bins. It uses the pd.cut function from pandas to achieve this.

110

top\_risk\_roads **=** road\_incident\_data**.**sort\_values(by**=**'severity\_score', ascending**=False**)**.**head(10)

print("\nTop 10 roads with highest severity scores:")

print(top\_risk\_roads[['road\_name', 'severity\_score', 'risk\_level']])

This code sorts the road\_incident\_data data frame by the severity\_score in descending order and selects the top 10 roads with the highest severity scores. It then prints these top 10 roads along with their severity\_score and risk\_level.

X2 **=** road\_incident\_data**.**drop(columns**=**['road\_name', 'total\_incidents', 'risk\_level'])

y2 **=** road\_incident\_data['risk\_level']

This code prepares the feature X2 and the target y2 for training a machine learning model, as it selects the relevant features and the target variable.

Dropping 'road\_name', 'total\_incidents', 'risk\_level' columns ensures that the model is trained on relevant, non-redundant features while avoiding data leakage and overfitting. This leads to a more accurate and generalizable model when predicting the risk level of roads based on incident data.

X2\_train, X2\_test, y2\_train, y2\_test **=** train\_test\_split(X2, y2, test\_size**=**0.2, random\_state**=**42)

Splitting the dataset into training and testing sets. The training set is used to train the machine learning model, and the testing set is used to evaluate the model's performance on unseen data.

111

rf2 **=** RandomForestClassifier(random\_state**=**42)

rf2**.**fit(X2\_train, y2\_train)

This code snippet trains a RandomForestClassifier using the training dataset.

113

y2\_pred\_test\_rf **=** rf2**.**predict(X2\_test)

print("RandomForest Test Accuracy for Risk Level Classification:", accuracy\_score(y2\_test, y2\_pred\_test\_rf))

print("RandomForest Test Classification Report for Risk Level Classification:\n", classification\_report(y2\_test, y2\_pred\_test\_rf))

evaluates the trained RandomForestClassifier model on the test dataset by predicting the target values and calculating performance metrics such as accuracy and classification report.

Iteration 3 include SMOT  
  
smote = SMOTE(random\_state=42, k\_neighbors=3)

X\_resampled, y\_resampled = smote.fit\_resample(X, y)  
  
This code makes the data set balanced per class