Brief

Steps taken to solve this case study are that of any end to end data science pipeline. Multiple insights about data were found during EDA. Next steps followed deciding on model , designing features for the same and training evaluation of model.

Steps

EDA

1. Imbalance in class labels
2. Long tailed distributions in multiple features were observed
3. Interesting contour plots were seen with Weather variables and class labels

Model selection and feature selection

1. Most of the features were hierachical in natue or categorical in nature
2. This type of data is well modelled by decision trees
3. Decision trees are light weight in nature and interpretable
4. Hence used decision trees for modelling

Feature engineering and transformations

1. Cleaning various columns such as `Wind_Direction` was done
2. Cleaning text data `Description` was done using various regex filters
3. In order for featurize `Description`
   A. Keywords were found using keyword extractor
   B. topk keywords were chosen to generate a binary vector
   C. occurence of keyword in the description would make the binary vector of respective dimension as 1
4. Featurizing `Zipcode`
   A. As zip codes are hierarchical in nature
   B. Each part of zip code marks a region
   C. Zip code was split into 2 parts
   D. first 2 digits of zip code named as to `zip_02` to create a new feature
   E. next 3 digits of zip code named as `zip_25` to create a new feature
   F. number of digits in zip code could be one feature that was added
   G. is the zip code compound zip code like `1234-567` a boolean feature was added

1. As most of the variables were categorical in nature , category encoded them using
   A. baseN encoder
   B. binary encoder

C. ordinal encoder

complete set of encoding methods applied is as follows:

```
categorical_features = {
    "Source": "base_2",  # 3 unique values
    "Side": "base_2",  # 2 unique values
    "City": "base_4",  # 11895 unique values
    "County": "base_4",  # 1713 unique values
    "State": "base_2",  # 49 unique values
    "Timezone": "base_2",  # 4 unique values
    "Airport_Code": "base_4",  # 2001 unique values
    "Wind_Direction": "base_4",  # 24 unique values (after
cleaning)
    "Weather_Condition": "base_4",  # 127 unique values
    "Sunrise_Sunset": "base_2",  # 2 unique values
    "Civil_Twilight": "base_2",  # 2 unique values
    "Nautical_Twilight": "base_2",  # 2 unique value
    "Astronomical_Twilight": "base_2",  # 2 unique value
    #
    # engineered features
    "zip_02": "ordinal",
    "zip_25": "ordinal",
}
```

Model training and evaluation

1. Data was fitted using decison trees
2. used only one hyperparameter for tuning : `max_depth`
3. Evaluation was done using:
   - Precision recall matrix (derived from confusion matrix)
   - log_loss (metric)
4. Model performed well on training and cross validation dataset but poorly performed on test set
5. One major reason for this behaviour could be - distributions of class labels in train and test set:

   ```
   plain_text
    class label counts of test set
    2    379695
    3    111298
    1     28205
    4     19989
    Name: Severity, dtype: int64

    class label counts of train set
    2    1993515
    3     887615
    4      92331
    1        969
    Name: Severity, dtype: int64
   ```
6. Class weights added during training did not work well during test set

Interpretation of results

Interpretation of predictions of this manner was done using external library:

````
```plain_text
Predicted class  [3]
Actual class  [3]

Path taken:

zip_02_0 < 46.5
0.5 <= kw_Northbound
kw_Hwy < 0.5
0.5 <= Airport_Code_6
Airport_Code_10 < 0.5
City_0 < 0.5
City_4 < 0.5
Side_0 < 0.5
0.5 <= Source_0
Distance(mi) < 1.41
Traffic_Signal < 0.5
zip_len < 7.5
```
````

Also feature importances were calculated in the end.

Future steps

1. Use ensemble methods to model the data and see the impact on log loss
2. Improve feature set by encoding text in a better format (text2vec)
3. Regularly train model on latest data so that model is robust to class imbalance