Report: Car Accident Severity

Introduction:

The purpose of this project is to warn user/people given the weather and the road conditions about the possibility of user/people getting into a car accident and how severe it would be, so that user/people would drive more carefully or even change travel if possible. It is an effort to reduce the frequency of car collisions which can save lives and damages happen to the common people due to car accidents.

Data understanding and preparation:

For this project we will be using a dataset available under below location.

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

We will be using 'SEVERITYCODE' as our predictor or target variable because it is used measure the severity of an accident from 0 to 5 within the dataset.

Feature Selection:

The main attributes used to weigh the severity of an accident from the given dataset are 'WEATHER', 'ROADCOND' and 'LIGHTCOND' as these attributes can be contributing factors to accidents. As these attributes type is “object” need to convert or create new attributes out of them with type as “int”, we will us label encoding to convert attributes in the desired type.

```
WEATHER          object
ROADCOND         object
LIGHTCOND        object
SEVERITYCODE      int64
WEATHER_le        int32
ROADCOND_le       int32
LIGHTCOND_le      int32
dtype: object
```

Balancing the Dataset :

Also the given dataset in not balanced, to balance it we can either down-sample majority class or up-sample minority class in the dataset, we will down-sample the dataset based on the SEVERITYCODE target column.

```
2    58188
1    58188
Name: SEVERITYCODE, dtype: int64
```

Below is the head of the converted dataset.

	WEATHER	ROADCOND	LIGHTCOND	SEVERITYCODE	WEATHER_le	ROADCOND_le	LIGHTCOND_le
0	Raining	Wet	Dark - Street Lights On	1	6	8	2
1	Clear	Dry	Daylight	1	1	0	5
2	Unknown	Unknown	Unknown	1	10	7	8
3	Clear	Dry	Daylight	1	1	0	5
4	Clear	Dry	Daylight	1	1	0	5

Methodology

As the dataset is pre-processed and prepared after selected, engineered, normalized, split into testing/training data & balanced, let us start training the model. The goal is to find the best machine learning model for predicting accident severity, we'll train different types of models and then compare the results to see which model works best for this task.

In this section we'll use the data prepared above to train these 4 different supervised machine learning classification algorithms:

- Decision Tree

- K Nearest Neighbor (KNN)

- Logistic Regression

- Support Vector Machine

Wwe'll then use different evaluation metrics like metric accuracy, F1-score and Log Loss to compare the results.

Data Normalization:

```
# normalize dataset
from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit(X).transform(X)
X[:2], y[:2]
```

```
(array([[ 1.13292854,  1.50336927, -1.42829196],
        [-0.71886902, -0.69313751,  0.39656901]]),
 array([1, 1], dtype=int64))
```

Train/Test Split:

We will use 30% of data for testing and rest 70% for training.

```
from sklearn.model_selection import train_test_split
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, random_state=3)
```

Machine Learning Models:

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(criterion="entropy", max_depth = 5)
dtree
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

```
dtree.fit(X_trainset, y_trainset)
predictionTree = dtree.predict(X_testset)
print(predictionTree[0:5])
print(y_testset[0:5])
```

```
[1 2 2 1 2]
[1 1 1 1 2]
```

K-Nearest-Neighbors

```
from sklearn.neighbors import KNeighborsClassifier
k = 11
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_trainset,y_trainset)
neigh
```

```
yhat = neigh.predict(X_testset)
yhat[0:5]
```

```
array([2, 2, 2, 1, 1], dtype=int64)
```

Logestic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_trainset,y_trainset)
LR
```

```
LogisticRegression(C=0.01, solver='liblinear')
```

```
yhatlr = LR.predict(X_testset)
yhatlr
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

```
yhat_prob = LR.predict_proba(X_testset)
yhat_prob
```

```
array([[0.53078711, 0.46921289],
       [0.51060032, 0.48939968],
       [0.51060032, 0.48939968],
       ...,
       [0.51060032, 0.48939968],
       [0.51060032, 0.48939968],
       [0.51060032, 0.48939968]])
```

SVM

```
from sklearn import svm
clf = svm.SVC(kernel='rbf')
clf.fit(X_trainset, y_trainset)
```

```
SVC()
```

```
yhatsvm = clf.predict(X_testset)
yhatsvm [0:5]
```

```
array([1, 2, 2, 1, 2], dtype=int64)
```

Results:

Results :

```
report_data = {'KNN': [knn_accuracy, knn_f1, 'NA'],
               'Decision Tree': [dt_accuracy, dt_f1, 'NA'],
               'SVM': [svm_accuracy, svm_f1, 'NA'],
               'Logistic Regression': [lr_accuracy, lr_f1, logloss]}
report_df = pd.DataFrame.from_dict(report_data, columns=['Accuracy', 'F1-score', 'LogLoss'], orient='index')
report_df.to_csv('results.csv')
report_df = pd.read_csv('results.csv', index_col=0)
report_df
```

	Accuracy	F1-score	LogLoss
KNN	0.540715	0.513749	NaN
Decision Tree	0.557319	0.531059	NaN
SVM	0.560169	0.531707	NaN
Logistic Regression	0.530753	0.518190	0.683575

The **K Nearest Neighbour**, **Decision Tree** and **Support Vector Machine** models performed significantly better than the Log Regression model and they achieved a Accuracy Score of 0.55 and a F1-Score of 0.53. The SVM model shows minimal better results, but these come at the cost of a much longer training time.

The **Logistic Regression** models performed with a accuracy Score of 0.526 and a F1-Score of 0.512 with LogLoss of 0.683, significantly worse than the other algorithms. The other model shows minimal better results, but also these results come at the cost of a much longer training time than the Logistic Regression Model.

Discussion:

As the initial dataset contains the categorical data with the type of "object". We cannot use this data type to proceed further in ML models hence we have used label encoding to create new attributes from these categorical attributes that were numerical in type such as int and float.

The other issue that the dataset had is the imbalance of the target attributes, using imbalance dataset could skew our result towards the majority class hence it would not be appropriate to use imbalance dataset. The imbalance dateset could be balanced using downsampling or upsampling, we have used sklearn's downsampling approach to generate working dataset.

After analyzing and cleaning the data, applied four ML classification models; K-Nearest Neighbor, Decision Tree, Logistic Regression and Support Vector Machine SVM. Although the other three are ideal for this project, we've used logistic regression as the target attribut is binary nature.

Evaluation metrics used to test the accuracy of our models were metric accuracy, jaccard index, f-1 score and (logloss for logistic regression). Evaluating best possible value for k, helped to improve our accuracy to be the best possible.

Conclusion:

The aim and purpose of the project is to predict the severity of a traffic accident based on historical data using machine learning classification model.

Based on historical data from weather conditions, road conditions, light conditions pointing to certain classes, we can conclude that these three conditions have a impact on whether or not travel could result in property damage (severity 1) or injury (severity 2).

Decision Tree algorithm was best suited for this task in most cases: My Decision Tree Model achieved a accuracy Score of 0.557 and a F1-Score of 0.5310 with optimal training times.

Other two models **K Nearest Neighbour** and **Support Vector Machine* model achieved almost same evaluation scores but at the cost of significantly higher training times.

The **Logistic Regression** models performed with a accuracy Score of 0.526 and a F1-Score of 0.512 with Log Loss of 0.683, significantly worse than the other algorithms.

The use of the outcome or prediction of this project in a real world, production environment could be challenged however it gives the basic understanding of the severity of the car accident under the provided circumstances.