COURSERA CAPSTONE
PROJECT

PREDICTING SEVERITY OF ROAD ACCIDENTS

NITESH ALONEY

INTRODUCTION

Road accidents area a very frequent talk. The main reasons for can be the carelessness of the drivers, the condition of the road where the accident had taken place, the weather conditions, Light conditions at the time of the accident. The land features, the unevenness of the road. These road accidents can of various severity, some accidents might cause damage, while some might cost your life.

And we can’t ignore that road accident death and injury rate has been always increasing. Since we have Technology like Data sciences and Machine Learning this issue can be tackled building models to predict the severity of the accidents.

PROBLEM STATEMENT

This demands a need for a technology/ algorithm using which you can predict the severity of the accident so that you can act accordingly to deal with that situation.

For Eg: consider a situation, where an accident happened, and it was very dangerous (high severity level) then ambulance can be called, and further medications can be carried out. on the other hand, if it was a mild collision that happened (low severity level) we can manage with the Band-Aids.

INTEREST

The Traffic Police department and Health care can collaboratively start on this technology/ algorithm. This task or problem statement can be tackled by data science methodologies. With this we can decrease the fatality rates significantly.

DATA PREPARATION

The Data-Collision dataset provided to us consists of data of the accidents and their severity based on various parameters. In total the dataset has 37 features or parameters and there are over 170000+ records(rows).

The columns which can contribute to the analysis are 'SEVERITYCODE', 'ADDRTYPE', 'ROADCOND', 'LIGHTCOND', 'WEATHER' which represent 'severity of the accident', 'address type that is kind of the location where the accident took place', 'road conditions', 'light conditions' and 'weather conditions' respectively, which is why we extract these columns from the dataset and overlook other for our analysis.

rest of the columns in the dataset are immaterial as they contain metadata like latitude, longitude, IDs, Direction of vehicles etc... let's keep them aside as of now because this data doesn't contribute much to our data analysis.

DATA PREPARATION

```
road_cond = df_use['ROADCOND'].unique()
light_cond = df_use['LIGHTCOND'].unique()
weather_cond = df_use['WEATHER'].unique()
addrType = df_use['ADDRTYPE'].unique()

print(road_cond, light_cond, weather_cond, addrType, sep='\n\n')

df_use = df_use[df_use['ROADCOND'].isin(road_cond)]
df_use = df_use[df_use['LIGHTCOND'].isin(light_cond)]
df_use = df_use[df_use['WEATHER'].isin(weather_cond)]
df_use = df_use[df_use['ADDRTYPE'].isin(addrType)]
```

```
['Wet' 'Dry' 'Snow/Slush' 'Ice' 'Other' 'Sand/Mud/Dirt' 'Standing Water'
 'Oil']

['Daylight' 'Dark - Street Lights On' 'Dark - No Street Lights' 'Dusk'
 'Dawn' 'Dark - Street Lights Off' 'Other' 'Dark - Unknown Lighting']

['Overcast' 'Raining' 'Clear' 'Snowing' 'Other' 'Fog/Smog/Smoke'
 'Sleet/Hail/Freezing Rain' 'Blowing Sand/Dirt' 'Severe Crosswind'
 'Partly Cloudy']

['Intersection' 'Block' 'Alley']
```

```
#dropping unknown values
index = df_use[df_use['ROADCOND']=='Unknown'].index
df_use = df_use.drop(index=index)

index = df_use[df_use['LIGHTCOND']=='Unknown'].index
df_use = df_use.drop(index=index)

index = df_use[df_use['WEATHER']=='Unknown'].index
df_use = df_use.drop(index=index)

index = df_use[df_use['ADDRTYPE']=='Unknown'].index
df_use = df_use.drop(index=index)

#dropping the null values
df_use = df_use.dropna()

del(index)
```

DATA CLEANING

DATA PREPARATION

DATA ENCODING

Now we've to change the categorical values to the numerical values since non numerical values can't be processed.

The following code segment transforms the categories to numbers and append the transformed columns to the dataset.

```
df_use['addr_cat'] = en.fit_transform(df_use['WEATHER'])
df_use['road_cat'] = en.fit_transform(df_use['ROADCOND'])
df_use['light_cat'] = en.fit_transform(df_use['LIGHTCOND'])
df_use['weather_cat'] = en.fit_transform(df_use['WEATHER'])

df_use.head()
```

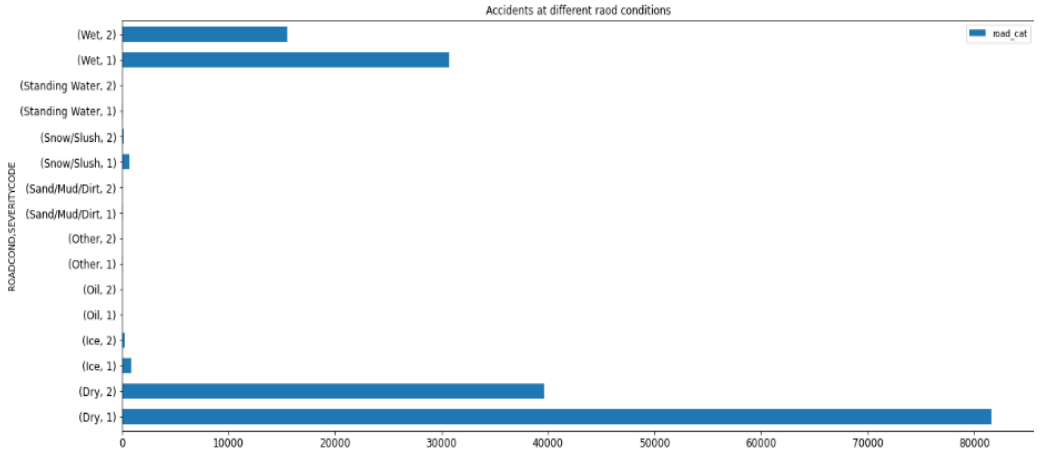
]:

	SEVERITYCODE	ADDRTYPE	ROADCOND	LIGHTCOND	WEATHER	addr_cat	road_cat	light_cat	weather_cat
0	2	Intersection	Wet	Daylight	Overcast	4	7	5	4
1	1	Block	Wet	Dark - Street Lights On	Raining	6	7	2	6
2	1	Block	Dry	Daylight	Overcast	4	0	5	4
3	1	Block	Dry	Daylight	Clear	1	0	5	1
4	2	Intersection	Wet	Daylight	Raining	6	7	5	6

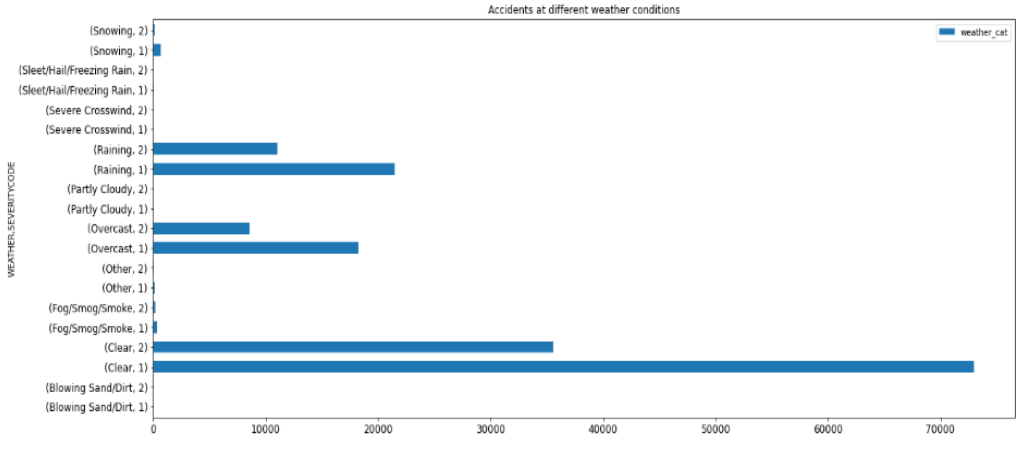
Here we used *'LabelEncoder'* to transform our data and 'en' variable is object of the same.

VISUALIZATIONS

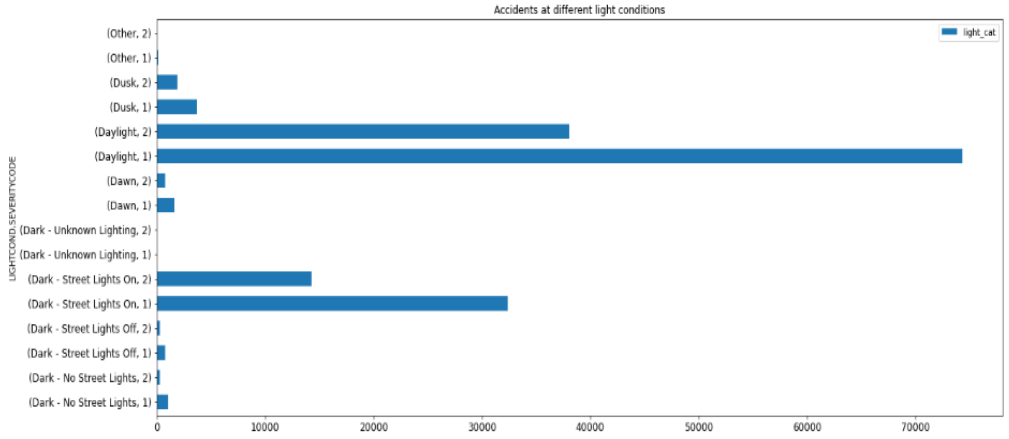
- **(Road Conditions, severity) vs (number of accidents)**



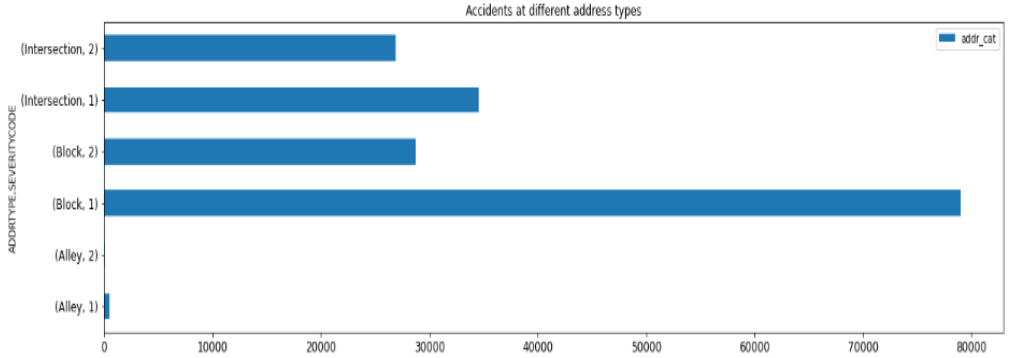
(Weather Conditions, severity) vs (number of accidents)



(Light Conditions, severity) vs (number of accidents)



(Address Type, severity) vs (number of accidents)



TRAIN - TEST DATA

Split the dataset into dependent (target) and independent variables X, Y

```
X, Y = data[['addr_cat','road_cat','light_cat','weather_cat']], data['SEVERITYCODE']
```

Split the dataset into training and testing data randomly using train_test_split function by scikit-learn

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.15, random_state=42)
```

CLASSIFIERS & EVALUATION METRICS

CLASSIFIERS

1. Decision Tree Classifier

2. K-Neighbors Classifier (With best k possible : 22 in our case)

3. Logistic Regression

EVALUATION metrics

- F_1-Score
- Accuracy Score
- Jaccard Score
- Log Loss (in case of logistic regression)

TRAIN - TEST DATA

Split the dataset into dependent (target) and independent variables X, Y

```
X, Y = data[['addr_cat','road_cat','light_cat','weather_cat']], data['SEVERITYCODE']
```

Split the dataset into training and testing data randomly using train_test_split function by scikit-learn

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.15, random_state=42)
```

```
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(144313, 4) (25468, 4) (144313,) (25468,)
```

TRAIN - TEST DATA

Split the dataset into dependent (target) and independent variables X, Y

```
X, Y = data[['addr_cat','road_cat','light_cat','weather_cat']], data['SEVERITYCODE']
```

Split the dataset into training and testing data randomly using train_test_split function by scikit-learn

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.15, random_state=42)
```

```
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(144313, 4) (25468, 4) (144313,) (25468,)
```

RESULTS

All the above used classifiers produce the same result. so let's rule of K-Nearest-Neighbors Classifier with 'k=22' as it takes lot of time to train the model using this. Now we're left with logistic Regression and Decision Trees Classifiers. both of these produce an 'F1_SCORE' of 0.8. But Logistic Regression is slightly better if you look at the figure below.

]:

	F1-Score	accuracy_score	jaccard_score	Logg_loss
Logistic Regeression	0.802802	0.670567	0.670567	0.632744
Decision Tree	0.802690	0.670488	0.670411	NaN
K-NeighScores (22)	0.802052	0.669782	0.669522	NaN

DISCUSSIONS

- we need not preprocess the dataset as all the values fall into the same range of under 10, which is why we preprocessing doesnt make much difference.
- we can also use some other features into consideration like number of vehicles, number of people injured. This features might also be helpful in determining the severity of the accident.
- Using neural networks concepts a model can be developed which can be be very efficient than what it is now.

CONCLUSION

We'll use Logistic regression as our final classifier as it is a bit better than the other classifiers with the f_1_score and accuracy score.

	F1-Score	accuracy_score	jaccard_score	Logg_loss
Logistic Regeression	0.802802	0.670567	0.670567	0.632744

For the given model Logistic Regression is best classifier choice for the given dataset.

THANK YOU