ACCIDENTS CLASSIFIER

ABSTRACT

WHAT THE PROBLEM IS ABOUT?

- Road accidents happen all the time due to several different conditions. An accident can be labeled as lethal or not through an index called "Severity Index"

SEVERITY INDEX

- It is a numerical integer index whose range can vary depending on the country.

OBJECTIVE

- Building a model that will be able to predict the severity index of an accident when given in input the road conditions and information

DATASET DESCRIPTION

2021

2020

2019

2018

2017

2016

6 YEARS

The information in the dataset were collected in the arc of 6 years in 49 countries in the US. The Severity Index goes from 1 to 4.

Number of records = about 2.9 million
Number of features = 47

The features include:

- Timestamp of the accident (so also if it was day or night)
- Road information (presence of signals, railway, junction, bump ecc...)
- Weather Condition (rain precipitation,wind speed, humidity ecc...)
- Brief description of the accident

FEATURE EXTRACTION

As said in the previous slide, one of the features represent the timestamp of the accident. The format of the timestamp makes it unusable for classification purposes, so some transformations were carried out on it.

STEP 1

Some of the timestamps contain also the milliseconds. The split() method was used to retrieve the part without milliseconds

STEP 2

transform all the timestamps according to the format %Y-%m-%d %H:%M:%S

STEP 3

Use the weekday() method on the timestamps to retrieve the day of the week the accident happened

STEP 4

Transform the integer values in the correspondent labels

```
({'monday': 419821,
  'tuesday': 443968,
  'wednesday': 455037,
  'thursday': 463477,
  'friday': 492074,
  'saturday': 311691,
  'sunday': 259274})
```

PREPROCESSING

Given the number of features, an inspection was carried out to understand if some of the columns were useless regarding our objective.

FEATURE DROP

Here on the right side there are examples of feature that were dropped

USELESS FEATURES

- ID = the id given to the accident
- ZIPCODE = the zipcode of the city
- DESCRIPTION = the description of the accident

REDUNDANT FEATURES

- CIVIL TWILIGHT,NAUTICAL TWILIGHT,ASTRONOMICAL TWILIGHT = different ways of considering the start of the sunset period
- END_TIME = timestamp of the end of the accident

FEATURES WITH ALL THE SAME VALUE

Some of the features have basically the same value for every record, so they are useless when it comes to classification. These features are:

- TRAFFIC CALMING
- TURNING LOOP
- ROUNDABOUT
- BUMP
- NO EXIT

PREPROCESSING

After all the useless features have been dropped, the following step consists in handling the missing values.

A total of 9 features were detected to have NaN values.

```
accidents_copy.isna().sum()
✓ 0.2s
```

Severity	0
Start_Time	0
Start_Lat	0
Start_Lng	0
End_Lat	0
End_Lng	0
Distance(mi)	0
Side	0
Temperature(F)	15420
Wind_Chill(F)	351617
Humidity(%)	16059
Pressure(in)	13027
Visibility(mi)	16547
Wind_Speed(mph)	82551
Precipitation(in)	385856
Weather_Condition	16736
Amenity	0
Crossing	0
Junction	0
Railway	0
Station	0
Stop	0
Traffic_Signal	0
Sunrise_Sunset	24

DROP OF THE ROWS

- TEMPERATURE
- HUMIDITY
- SUNRISE_SUNSET

MEAN OF THE FEATURE

- WIND_CHILL
- PRESSURE
- VISIBILITY
- WIND_SPEED
- PRECIPITATION

MODE OF THE FEATURE

- WEATHER_CONDITION

PREPROCESSING

RESULT

- After the preprocessing steps of the previous slides that were carried out, the dataset contained about 2.5 million rows and 24 features.

SEVERITY INDEX

- Upon futher inspection of the Severity feature, it could be observed that the classes associated with the values 1 and 4 were very few, infact the latter was only present in 26000 records, while the classes for 2 and 3 were more present in the dataset. In this sense undersampling and oversampling didn't produce good results if 1 and 4 were taken into account, but if all the rows related to them were dropped, much better output were produced, so the related rows were dropped to keep the problem interesting.

FURTHER ROWS DROPPED

- Even after these modifications, the number of rows was still over 2 million, which was too much in terms of computational time for the classification. The goal was to keep them under 1 million and for this purpose the date of the accident was exploited: as said in the beginning, the data were collected in the arc of 6 years, so dropping rows related to some of the years was enough.

PREPROCESSING

The only thing that remained to handle were the categorical features. More specifically, there were 2 feature of this type: START_TIME and WEATHER_CONDITION

ENCODING

Among the possible strategies, the best one was observed to be the get_dummies() method

GET DUMMIES

RESULT

The original 2 features were dropped and replaced by a column for each of the categorical value

CROSS VALIDATION

To ensure the quality of the eventual model and compare the performance of different classifiers, a cross validation was carried out.
Pipeline and pipeline estimators were used to do so in order to prevent Data Leakage and therefore getting clean results. Oversampling (SMOTE) in combination with UnderSampling (RandomUnderSampling) was used to get a better result.

ADABOOST

Fscore: 82%

LOGISTIC REGRESSION

Fscore: 80%

RANDOM FOREST

Fscore: 86%

K-NN

Fscore: 82%

```
y = preprocessed_accidents_df['Severity']
X = preprocessed_accidents_df.drop('Severity',axis=1)

pipeline_estimators = [('scaling',StandardScaler()),('clf',classifier)]
pipe = Pipeline(pipeline_estimators)
skf = StratifiedKFold(10, shuffle = True, random_state = 21)

results_validation = cross_validate(pipe,
                                    X,
                                    y,
                                    scoring = {'fscore': make_scorer(f1_score),
                                               'accuracy': make_scorer(accuracy_score)},
                                    error_score= 'raise',
                                    return_estimator = True,
                                    cv = skf,
                                    n_jobs = -1)

metrics = results_validation['test_fscore']

print(results_validation)
print("Mean fscore: ",mean(metrics))
```

CROSS VALIDATION

Here the best accuracy results are reported with some of the possible combination

DecisionTree

Criterion:

- GINI -> 79%
- ENTROPY -> 80%

Adaboost

n_estimators:

- 50 -> 81,9%
- 60 -> 82%
- 70 -> 82,3%
- 80 -> 82,7%

K-NN

k:

- 10 -> 81,9%
- 9 -> 82%
- 8 -> 82%
- 7 -> 82%
- 6 -> 82%
- 5 -> 82%

RandomForest

Criterion:

- GINI -> 85,8%
- ENTROPY -> 86%

Logistic Regression

Class_Weight:

- NONE-> 80%
- BALANCED-> 80%

TRAINING THE MODEL

Since the goal was building a model able to predict the severity of an accident, the dataset was split beforehand in train data and test data. The preprocessing steps were carried out and the best result from cross-validation was used to create the model

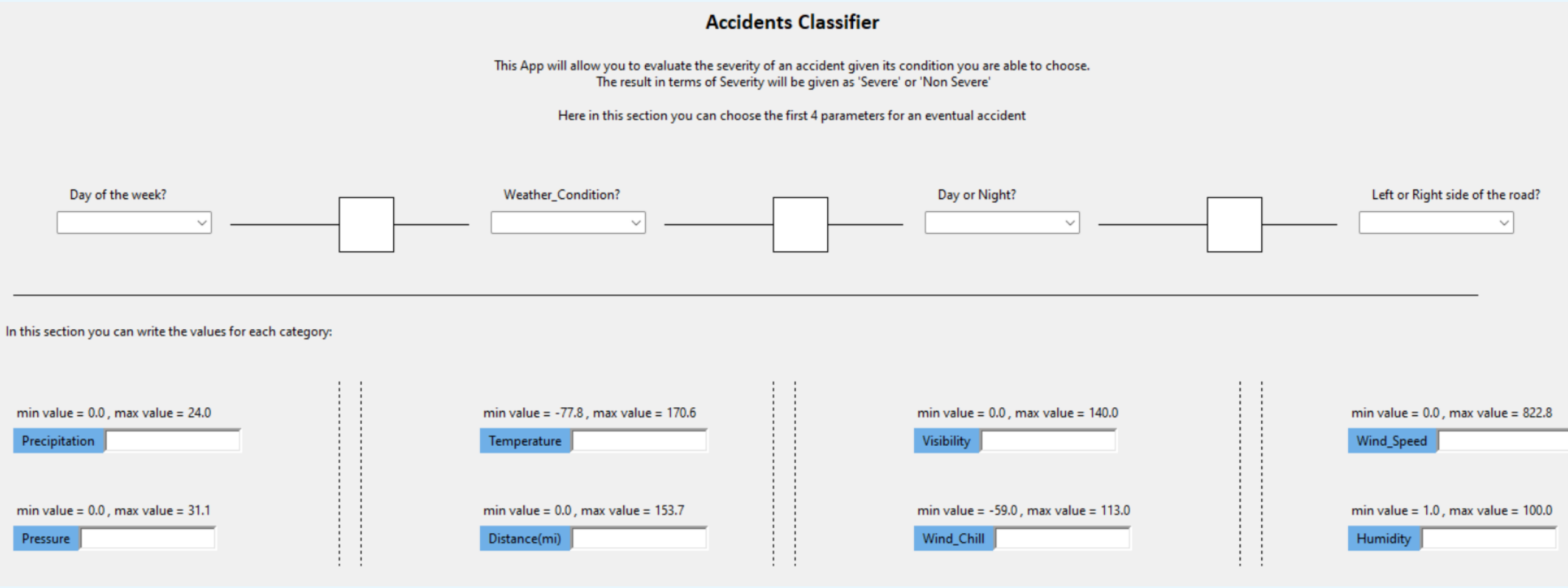
SAVING THE MODEL

Once the training was completed the model was saved in order to be imported for creating the Application.

```python
import pickle

class MyClass():
    def __init__(self, param):
        self.param = param

def save_object(obj,filename):
    try:
        with open(filename, "wb") as f:
            pickle.dump(obj, f, protocol=pickle.HIGHEST_PROTOCOL)
    except Exception as ex:
        print("Error during pickling object (Possibly unsupported):", ex)

save_object(X_train,"X_train.pickle")
save_object(X_test,"X_test.pickle")
save_object(y_train,"y_train.pickle")
save_object(y_test,"y_test.pickle")
```

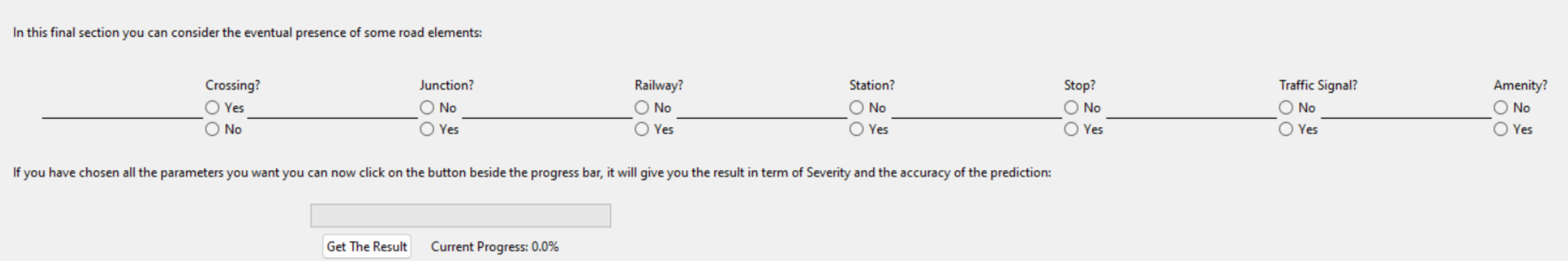
```python
import pickle

class MyClass():
    def __init__(self, param):
        self.param = param

def load_object(filename):
    try:
        with open(filename, "rb") as f:
            return pickle.load(f)
    except Exception as ex:
        print("Error during unpickling object (Possibly unsupported):", ex)

X_train = load_object("X_train.pickle")
X_test = load_object("X_test.pickle")
y_train = load_object("y_train.pickle")
y_test = load_object("y_test.pickle")
```

APPLICATION



- **First Section**: Comboboxes to choose the option
- **Second Section**: the user can type the values into the Entries

APPLICATION



- **Third Section**: Radio buttons to choose the option
- **Last Section**: Progress bar and related button to compute the prediction

Once the prediction has been computed, the result will be shown to the user.

PAPERS

There are 2 papers regarding this dataset and its classification problem. The model that were decided to use were mostly Logistic Regression (LR) and also Deep Neural Networks (DNN). As said, classification was carried also in this case and the precision of the models was measured with the Weighted Average F1 Score. The models were tested on the records regarding 6 cities, here is the result:

Model / City	LR	DNN
	W-Avg	W-Avg
Atlanta	0.83	0.83
Austin	0.87	0.87
Charlotte	0.83	0.82
Dallas	0.87	0.87
Houston	0.88	0.88
Los Angeles	0.78	0.75

- https://arxiv.org/pdf/1906.05409.pdf
- https://arxiv.org/pdf/1909.09638.pdf

THANKS FOR THE ATTENTION