Automobile Accident Severity Prediction

Given *data about accidents in the US*, let's try to pred[link text](https:// link text)ict the **severity** of a given accident.

We will use a TensorFlow ANN to make our predictions.

Getting Started

```
import numpy as np
import pandas as pd

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

import tensorflow as tf
```

```
data = pd.read_csv('../input/us-accidents/US_Accidents_June20.csv', nrows=400000)
```

```
data
```

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	...	Roundabout	Station
0	A-1	MapQuest	201.0	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	NaN	...	False	False
1	A-2	MapQuest	201.0	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	NaN	...	False	False
2	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	NaN	...	False	False
3	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	NaN	...	False	False
4	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	NaN	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
399995	A-400001	MapQuest	241.0	3	2017-04-25 11:53:42	2017-04-25 12:23:16	37.717747	-121.532150	NaN	NaN	...	False	False
399996	A-400002	MapQuest	201.0	3	2017-04-25 12:08:17	2017-04-25 12:37:47	37.932465	-122.403290	NaN	NaN	...	False	False
399997	A-400003	MapQuest	201.0	3	2017-04-25 12:06:21	2017-04-25 12:35:52	37.799576	-122.222092	NaN	NaN	...	False	False
399998	A-400004	MapQuest	201.0	2	2017-04-25 12:00:56	2017-04-25 12:29:00	37.009869	-121.515793	NaN	NaN	...	False	False
399999	A-400005	MapQuest	201.0	2	2017-04-25 12:06:54	2017-04-25 12:36:39	38.978897	-121.382561	NaN	NaN	...	False	False

400000 rows × 49 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400000 entries, 0 to 399999
Data columns (total 49 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   ID                     400000 non-null  object
 1   Source                 400000 non-null  object
 2   TMC                    400000 non-null  float64
 3   Severity               400000 non-null  int64
 4   Start_Time             400000 non-null  object
 5   End_Time               400000 non-null  object
 6   Start_Lat              400000 non-null  float64
 7   Start_Lng              400000 non-null  float64
 8   End_Lat                0 non-null       float64
 9   End_Lng                0 non-null       float64
 10  Distance(mi)           400000 non-null  float64
 11  Description            400000 non-null  object
 12  Number                 142925 non-null  float64
 13  Street                 400000 non-null  object
 14  Side                   400000 non-null  object
 15  City                   399981 non-null  object
 16  County                 400000 non-null  object
 17  State                  400000 non-null  object
 18  Zipcode                399954 non-null  object
 19  Country                400000 non-null  object
 20  Timezone               399954 non-null  object
 21  Airport_Code           399954 non-null  object
 22  Weather_Timestamp      396789 non-null  object
 23  Temperature(F)         394083 non-null  float64
 24  Wind_Chill(F)          59095 non-null   float64
 25  Humidity(%)            393489 non-null  float64
 26  Pressure(in)           395351 non-null  float64
 27  Visibility(mi)         391219 non-null  float64
 28  Wind_Direction         396768 non-null  object
 29  Wind_Speed(mph)        325825 non-null  float64
 30  Precipitation(in)      42047 non-null   float64
 31  Weather_Condition      391790 non-null  object
 32  Amenity                400000 non-null  bool
 33  Bump                   400000 non-null  bool
 34  Crossing               400000 non-null  bool
 35  Give_Way               400000 non-null  bool
 36  Junction               400000 non-null  bool
 37  No_Exit                400000 non-null  bool
 38  Railway                400000 non-null  bool
 39  Roundabout             400000 non-null  bool
 40  Station                400000 non-null  bool
 41  Stop                   400000 non-null  bool
 42  Traffic_Calming        400000 non-null  bool
 43  Traffic_Signal         400000 non-null  bool
 44  Turning_Loop           400000 non-null  bool
 45  Sunrise_Sunset         399981 non-null  object
 46  Civil_Twilight         399981 non-null  object
 47  Nautical_Twilight      399981 non-null  object
 48  Astronomical_Twilight  399981 non-null  object
dtypes: bool(13), float64(14), int64(1), object(21)
memory usage: 114.8+ MB
```

Missing Values

```
data.isna().mean()
```

```
ID                       0.000000
Source                   0.000000
TMC                      0.000000
Severity                 0.000000
Start_Time               0.000000
End_Time                 0.000000
Start_Lat                0.000000
Start_Lng                0.000000
End_Lat                  1.000000
End_Lng                  1.000000
Distance(mi)             0.000000
Description              0.000000
Number                   0.642687
Street                   0.000000
Side                     0.000000
City                     0.000048
County                   0.000000
State                    0.000000
Zipcode                  0.000115
Country                  0.000000
Timezone                 0.000115
Airport_Code             0.000115
Weather_Timestamp        0.008027
Temperature(F)           0.014793
Wind_Chill(F)            0.852263
Humidity(%)              0.016278
Pressure(in)             0.011622
Visibility(mi)           0.021952
Wind_Direction           0.008080
Wind_Speed(mph)          0.185438
Precipitation(in)        0.894883
Weather_Condition        0.020525
Amenity                  0.000000
Bump                     0.000000
Crossing                 0.000000
Give_Way                 0.000000
```

```
Junction                 0.000000
No_Exit                  0.000000
Railway                  0.000000
Roundabout               0.000000
Station                  0.000000
Stop                     0.000000
Traffic_Calming          0.000000
Traffic_Signal           0.000000
Turning_Loop             0.000000
Sunrise_Sunset           0.000048
Civil_Twilight           0.000048
Nautical_Twilight        0.000048
Astronomical_Twilight    0.000048
dtype: float64
```

```
null_columns = ['End_Lat', 'End_Lng', 'Number', 'Wind_Chill(F)', 'Precipitation(in)']
```

```
data = data.drop(null_columns, axis=1)
```

```
data.isna().sum()
```

```
ID                           0
Source                       0
TMC                          0
Severity                     0
Start_Time                   0
End_Time                     0
Start_Lat                    0
Start_Lng                    0
Distance(mi)                 0
Description                  0
Street                       0
Side                         0
City                        19
County                       0
State                        0
Zipcode                     46
Country                      0
Timezone                    46
Airport_Code                46
Weather_Timestamp         3211
Temperature(F)            5917
Humidity(%)               6511
Pressure(in)              4649
Visibility(mi)            8781
Wind_Direction            3232
Wind_Speed(mph)          74175
Weather_Condition         8210
Amenity                      0
Bump                         0
Crossing                     0
Give_Way                     0
Junction                     0
No_Exit                      0
Railway                      0
Roundabout                   0
Station                      0
Stop                         0
Traffic_Calming              0
Traffic_Signal               0
Turning_Loop                 0
Sunrise_Sunset              19
Civil_Twilight              19
Nautical_Twilight           19
Astronomical_Twilight       19
dtype: int64
```

```
data = data.dropna(axis=0).reset_index(drop=True)
```

```
print("Total missing values:", data.isna().sum().sum())
```

```
Total missing values: 0
```

```
data
```

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	Distance(mi)	Description	...	Roundabou
0	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	0.01	Accident on OH-32 State Route 32 Westbound at ...	...	Fals
1	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	0.01	Accident on I-75 Southbound at Exits 52 52B US...	...	Fals
2	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	0.01	Accident on McEwen Rd at OH-725 Miamisburg Cen...	...	Fals
3	A-6	MapQuest	201.0	3	2016-02-08 07:44:26	2016-02-08 08:14:26	40.100590	-82.925194	0.01	Accident on I-270 Outerbelt Northbound near Ex...	...	Fals
4	A-7	MapQuest	201.0	2	2016-02-08 07:59:35	2016-02-08 08:29:35	39.758274	-84.230507	0.00	Accident on Oakridge Dr at Woodward Ave. Expec...	...	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...
320976	A-400001	MapQuest	241.0	3	2017-04-25 11:53:42	2017-04-25 12:23:16	37.717747	-121.532150	0.01	One lane blocked due to accident on I-580 West...	...	Fals
320977	A-400002	MapQuest	201.0	3	2017-04-25 12:08:17	2017-04-25 12:37:47	37.932465	-122.403290	0.01	Right hand shoulder blocked due to accident on...	...	Fals
320978	A-400003	MapQuest	201.0	3	2017-04-25 12:06:21	2017-04-25 12:35:52	37.799576	-122.222092	0.01	Slow lane blocked due to accident on I-580 Wes...	...	Fals
320979	A-400004	MapQuest	201.0	2	2017-04-25 12:00:56	2017-04-25 12:29:00	37.009869	-121.515793	0.01	Turning lane blocked due to accident on CA-152...	...	Fals
320980	A-400005	MapQuest	201.0	2	2017-04-25 12:06:54	2017-04-25 12:36:39	38.978897	-121.382561	0.01	Accident on Riosa Rd both ways at CA-65.	...	Fals

320981 rows × 44 columns

Unnecessary Columns

```
{column: len(data[column].unique()) for column in data.columns if data.dtypes[column] == 'object'}
```

```
{'ID': 320981,
 'Source': 2,
 'Start_Time': 316629,
 'End_Time': 314439,
 'Description': 236513,
 'Street': 36206,
 'Side': 3,
 'City': 4023,
 'County': 548,
 'State': 28,
 'Zipcode': 57076,
 'Country': 1,
 'Timezone': 4,
 'Airport_Code': 638,
 'Weather_Timestamp': 78674,
 'Wind_Direction': 23,
 'Weather_Condition': 67,
 'Sunrise_Sunset': 2,
 'Civil_Twilight': 2,
 'Nautical_Twilight': 2,
 'Astronomical_Twilight': 2}
```

```
unneeded_columns = ['ID', 'Description', 'Street', 'City', 'Zipcode', 'Country']

data = data.drop(unneeded_columns, axis=1)
```

```
data
```

	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	Distance(mi)	Side	County	...	Roundabout	Sta
0	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	0.01	R	Clermont	...	False	I
1	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	0.01	R	Montgomery	...	False	I
2	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	0.01	R	Montgomery	...	False	I
3	MapQuest	201.0	3	2016-02-08 07:44:26	2016-02-08 08:14:26	40.100590	-82.925194	0.01	R	Franklin	...	False	I
4	MapQuest	201.0	2	2016-02-08 07:59:35	2016-02-08 08:29:35	39.758274	-84.230507	0.00	R	Montgomery	...	False	I
...	...	...	...	...	...	...	...	...	...	...	...	...	
320976	MapQuest	241.0	3	2017-04-25 11:53:42	2017-04-25 12:23:16	37.717747	-121.532150	0.01	R	San Joaquin	...	False	I
320977	MapQuest	201.0	3	2017-04-25 12:08:17	2017-04-25 12:37:47	37.932465	-122.403290	0.01	R	Contra Costa	...	False	I
320978	MapQuest	201.0	3	2017-04-25 12:06:21	2017-04-25 12:35:52	37.799576	-122.222092	0.01	R	Alameda	...	False	I
320979	MapQuest	201.0	2	2017-04-25 12:00:56	2017-04-25 12:29:00	37.009869	-121.515793	0.01	R	Santa Clara	...	False	I
320980	MapQuest	201.0	2	2017-04-25 12:06:54	2017-04-25 12:36:39	38.978897	-121.382561	0.01	R	Placer	...	False	I

320981 rows × 38 columns

```
def get_years(df, column):
    return df[column].apply(lambda date: date[0:4])

def get_months(df, column):
    return df[column].apply(lambda date: date[5:7])
```

```
data['Start_Time_Month'] = get_months(data, 'Start_Time')
data['Start_Time_Year'] = get_years(data, 'Start_Time')

data['End_Time_Month'] = get_months(data, 'End_Time')
data['End_Time_Year'] = get_years(data, 'End_Time')

data['Weather_Timestamp_Month'] = get_months(data, 'Weather_Timestamp')
data['Weather_Timestamp_Year'] = get_years(data, 'Weather_Timestamp')
```

```
data = data.drop(['Start_Time', 'End_Time', 'Weather_Timestamp'], axis=1)
```

```
data
```

	Source	TMC	Severity	Start_Lat	Start_Lng	Distance(mi)	Side	County	State	Timezone	...	Sunrise_Sunset	Civ
0	MapQuest	201.0	2	39.063148	-84.032608	0.01	R	Clermont	OH	US/Eastern	...	Night	
1	MapQuest	201.0	3	39.747753	-84.205582	0.01	R	Montgomery	OH	US/Eastern	...	Night	
2	MapQuest	201.0	2	39.627781	-84.188354	0.01	R	Montgomery	OH	US/Eastern	...	Day	
3	MapQuest	201.0	3	40.100590	-82.925194	0.01	R	Franklin	OH	US/Eastern	...	Day	
4	MapQuest	201.0	2	39.758274	-84.230507	0.00	R	Montgomery	OH	US/Eastern	...	Day	
...	...	...	...	...	...	...	...	...	...	...	...	...	
320976	MapQuest	241.0	3	37.717747	-121.532150	0.01	R	San Joaquin	CA	US/Pacific	...	Day	
320977	MapQuest	201.0	3	37.932465	-122.403290	0.01	R	Contra Costa	CA	US/Pacific	...	Day	
320978	MapQuest	201.0	3	37.799576	-122.222092	0.01	R	Alameda	CA	US/Pacific	...	Day	
320979	MapQuest	201.0	2	37.009869	-121.515793	0.01	R	Santa Clara	CA	US/Pacific	...	Day	
320980	MapQuest	201.0	2	38.978897	-121.382561	0.01	R	Placer	CA	US/Pacific	...	Day	

320981 rows × 41 columns

Encoding

```
def onehot_encode(df, columns, prefixes):
    df = df.copy()
    for column, prefix in zip(columns, prefixes):
        dummies = pd.get_dummies(df[column], prefix=prefix)
        df = pd.concat([df, dummies], axis=1)
        df = df.drop(column, axis=1)
    return df
```

```
{column: len(data[column].unique()) for column in data.columns if data.dtypes[column] == 'object'}
```

```
{'Source': 2,
 'Side': 3,
 'County': 548,
 'State': 28,
 'Timezone': 4,
 'Airport_Code': 638,
 'Wind_Direction': 23,
 'Weather_Condition': 67,
 'Sunrise_Sunset': 2,
 'Civil_Twilight': 2,
 'Nautical_Twilight': 2,
 'Astronomical_Twilight': 2,
 'Start_Time_Month': 12,
 'Start_Time_Year': 2,
 'End_Time_Month': 12,
 'End_Time_Year': 2,
 'Weather_Timestamp_Month': 12,
 'Weather_Timestamp_Year': 2}
```

```
data = onehot_encode(
    data,
    columns=['Side', 'County', 'State', 'Timezone', 'Airport_Code', 'Wind_Direction', 'Weather_Condition'],
    prefixes=['SI', 'CO', 'ST', 'TZ', 'AC', 'WD', 'WC']
)
```

```
data
```

	Source	TMC	Severity	Start_Lat	Start_Lng	Distance(mi)	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)	.
0	MapQuest	201.0	2	39.063148	-84.032608	0.01	36.0	100.0	29.67	10.0	
1	MapQuest	201.0	3	39.747753	-84.205582	0.01	35.1	96.0	29.64	9.0	
2	MapQuest	201.0	2	39.627781	-84.188354	0.01	36.0	89.0	29.65	6.0	
3	MapQuest	201.0	3	40.100590	-82.925194	0.01	37.9	97.0	29.63	7.0	
4	MapQuest	201.0	2	39.758274	-84.230507	0.00	34.0	100.0	29.66	7.0	
...	...	...	...	...	...	...	...	...	...	...	
320976	MapQuest	241.0	3	37.717747	-121.532150	0.01	60.1	55.0	30.09	10.0	
320977	MapQuest	201.0	3	37.932465	-122.403290	0.01	63.0	52.0	30.05	10.0	
320978	MapQuest	201.0	3	37.799576	-122.222092	0.01	63.0	54.0	30.11	10.0	
320979	MapQuest	201.0	2	37.009869	-121.515793	0.01	62.6	48.0	30.11	10.0	
320980	MapQuest	201.0	2	38.978897	-121.382561	0.01	64.4	49.0	30.05	10.0	

320981 rows × 1345 columns

```
def get_binary_column(df, column):
    if column == 'Source':
        return df[column].apply(lambda x: 1 if x == 'MapQuest' else 0)
    else:
        return df[column].apply(lambda x: 1 if x == 'Day' else 0)
```

```
data['Source'] = get_binary_column(data, 'Source')

data['Sunrise_Sunset'] = get_binary_column(data, 'Sunrise_Sunset')
data['Civil_Twilight'] = get_binary_column(data, 'Civil_Twilight')
data['Nautical_Twilight'] = get_binary_column(data, 'Nautical_Twilight')
data['Astronomical_Twilight'] = get_binary_column(data, 'Astronomical_Twilight')
```

```
data
```

	Source	TMC	Severity	Start_Lat	Start_Lng	Distance(mi)	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)	...
0	1	201.0	2	39.063148	-84.032608	0.01	36.0	100.0	29.67	10.0	...
1	1	201.0	3	39.747753	-84.205582	0.01	35.1	96.0	29.64	9.0	...
2	1	201.0	2	39.627781	-84.188354	0.01	36.0	89.0	29.65	6.0	...
3	1	201.0	3	40.100590	-82.925194	0.01	37.9	97.0	29.63	7.0	...
4	1	201.0	2	39.758274	-84.230507	0.00	34.0	100.0	29.66	7.0	...
...	...	...	...	...	...	...	...	...	...	...	...
320976	1	241.0	3	37.717747	-121.532150	0.01	60.1	55.0	30.09	10.0	...
320977	1	201.0	3	37.932465	-122.403290	0.01	63.0	52.0	30.05	10.0	...
320978	1	201.0	3	37.799576	-122.222092	0.01	63.0	54.0	30.11	10.0	...
320979	1	201.0	2	37.009869	-121.515793	0.01	62.6	48.0	30.11	10.0	...
320980	1	201.0	2	38.978897	-121.382561	0.01	64.4	49.0	30.05	10.0	...

320981 rows × 1345 columns

Splitting/Scaling

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

```
y.unique()
```

```
array([2, 3, 1, 4])
```

```
y = y - 1
X = X.astype(np.float)


scaler = StandardScaler()

X = scaler.fit_transform(X)


X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=100)
```

Training

```
X.shape
```

```
(320981, 1344)
```

```
inputs = tf.keras.Input(shape=(X.shape[1],))
x = tf.keras.layers.Dense(64, activation='relu')(inputs)
x = tf.keras.layers.Dense(64, activation='relu')(x)
outputs = tf.keras.layers.Dense(4, activation='softmax')(x)

model = tf.keras.Model(inputs, outputs)

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

batch_size = 32
epochs = 20

history = model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    batch_size=batch_size,
    epochs=epochs,
    callbacks=[
        tf.keras.callbacks.ReduceLROnPlateau(),
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=3,
            restore_best_weights=True
        )
    ]
)
```

```
Epoch 1/20
5618/5618 [==============================] - 12s 2ms/step - loss: 0.4421 - accuracy: 0.7921 - val_loss: 0.4166 - val_accuracy: 0.804
Epoch 2/20
5618/5618 [==============================] - 11s 2ms/step - loss: 0.4104 - accuracy: 0.8074 - val_loss: 0.4118 - val_accuracy: 0.808
Epoch 3/20
5618/5618 [==============================] - 11s 2ms/step - loss: 0.4000 - accuracy: 0.8113 - val_loss: 0.4062 - val_accuracy: 0.811
Epoch 4/20
5618/5618 [==============================] - 11s 2ms/step - loss: 0.3945 - accuracy: 0.8149 - val_loss: 0.4044 - val_accuracy: 0.811
Epoch 5/20
5618/5618 [==============================] - 11s 2ms/step - loss: 0.3898 - accuracy: 0.8163 - val_loss: 0.4047 - val_accuracy: 0.81
```