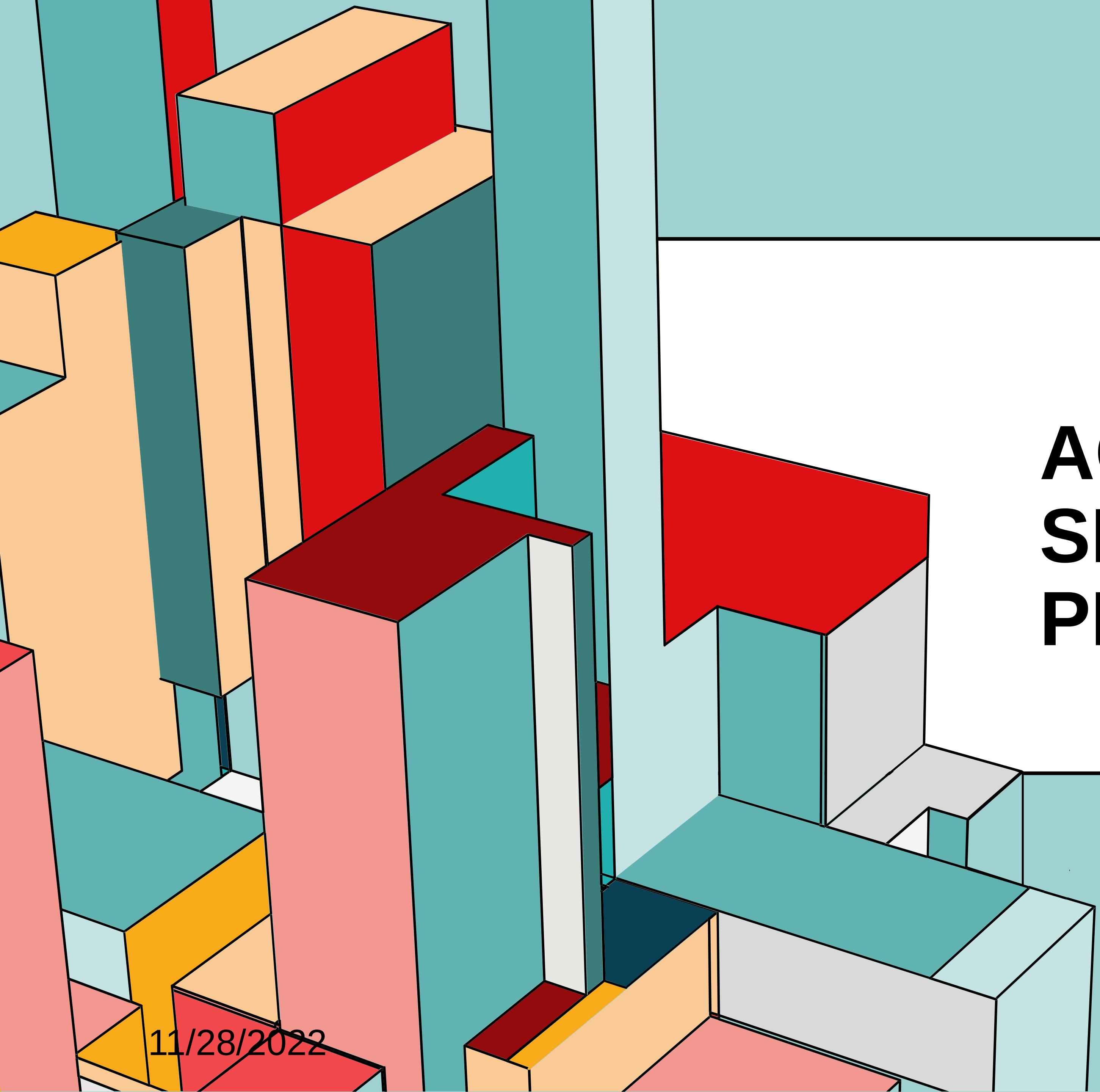


ACCIDENT SEVERITY PREDICTION

Kartik Lande

11/28/2022

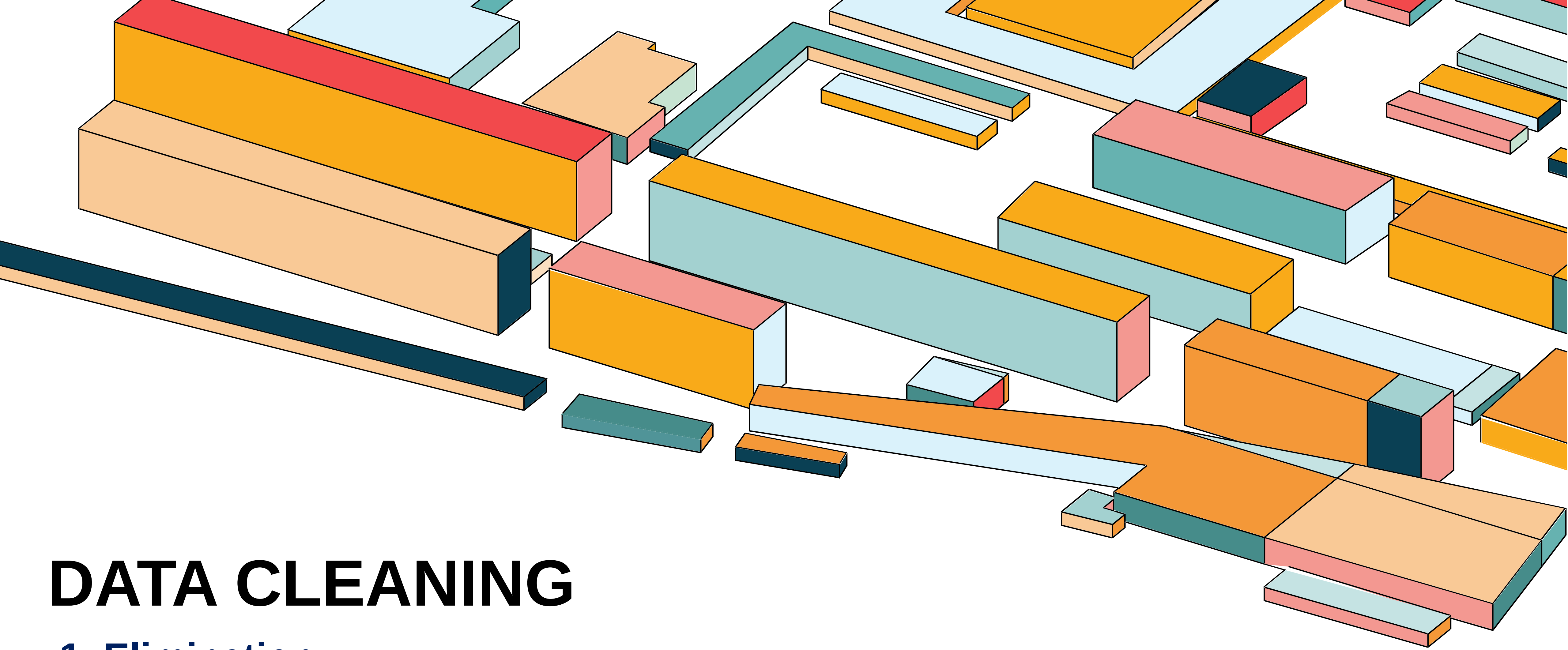
INTRODUCTION

2319 Accidents every day

US Accidents (2016 - 2021)

- This is a dataset on nationwide traffic accidents including data from 49 states in the US.
- It originally included 4.2 million accident records and 47 columns which we have reduced to 30 after data cleaning and preprocessing.
- The dataset includes information on factors like location, time, weather conditions and severity of an accident.
- We will be using these attributes to analyze the circumstances of the accidents, draw insights from it and create a ML model to predict the severity of an accident based on certain assumed attributes.

California



DATA CLEANING

1. Elimination
2. Simplification
3. Transformation

ELIMINATION

Handling columns with large number of empty values :

```
In [5]:  df.isna().sum()
```

```
Out[5]: ID                        0
Severity                  0
Start_Time                0
End_Time                  0
Start_Lat                 0
Start_Lng                 0
End_Lat                   0
End_Lng                   0
Distance(mi)              0
Description               0
Number              1743911
Street                    2
Side                      0
City                    137
County                    0
State                     0
Zipcode                1319
Country                   0
Timezone               3659
Airport_Code           9549
Weather_Timestamp     50736
Temperature(F)        69274
Wind_Chill(F)        469643
Humidity(%)           73092
Pressure(in)          59200
Visibility(mi)        70546
Wind_Direction        73775
Wind_Speed(mph)      157944
Precipitation(in)    549458
Weather_Condition     70636
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         2867
Civil_Twilight         2867
Nautical_Twilight      2867
Astronomical_Twilight  2867
dtype: int64
```

ELIMINATION

Dropping columns which have only one class :

```
In [9]:  cat_names = ['Country', 'Timezone', 'Bump', 'Crossing',
                      'Junction', 'No_Exit', 'Railway', 'Roundabout', 'Station',
                      'Stop', 'Traffic_Signal', 'Turning_Loop', 'Sunrise_Sunset']
         print("Unique count of categorical features:")
         for i in cat_names:
             print(i,df[i].unique().size)
```

```
Unique count of categorical features:
Country 1
Timezone 5
Bump 2
Crossing 2
Junction 2
No_Exit 2
Railway 2
Roundabout 2
Station 2
Stop 2
Traffic_Signal 2
Turning_Loop 1
Sunrise_Sunset 3
```

SIMPLIFICATION

Wind Direction :

In [11]:
```python
print("Wind Direction: ", df['Wind_Direction'].unique())
```

```
Wind Direction:  ['SW' 'Calm' 'WSW' 'WNW' 'West' 'NNW' 'South' 'W' 'NW' 'North' 'SSE' 'SSW'
 'ESE' 'SE' nan 'East' 'Variable' 'NNE' 'NE' 'ENE' 'CALM' 'S' 'VAR' 'N'
 'E']
```

In [12]:
```python
df.loc[df['Wind_Direction']=='Calm','Wind_Direction'] = 'CALM'
df.loc[(df['Wind_Direction']=='West')|(df['Wind_Direction']=='WSW')|(df['Wind_Direction']=='WNW'),'Wind_Direction'] = 'W'
df.loc[(df['Wind_Direction']=='South')|(df['Wind_Direction']=='SSW')|(df['Wind_Direction']=='SSE'),'Wind_Direction'] = 'S'
df.loc[(df['Wind_Direction']=='North')|(df['Wind_Direction']=='NNW')|(df['Wind_Direction']=='NNE'),'Wind_Direction'] = 'N'
df.loc[(df['Wind_Direction']=='East')|(df['Wind_Direction']=='ESE')|(df['Wind_Direction']=='ENE'),'Wind_Direction'] = 'E'
df.loc[df['Wind_Direction']=='Variable','Wind_Direction'] = 'VAR'
print("Wind Direction after simplification: ", df['Wind_Direction'].unique())
```

```
Wind Direction after simplification:  ['SW' 'CALM' 'W' 'N' 'S' 'NW' 'E' 'SE' nan 'VAR' 'NE']
```

SIMPLIFICATION

Weather Condition :

```
Weather Conditions:  ['', 'Clear', 'Cloudy', 'Drifting Snow', 'Drizzle', 'Dust', 'Dust Whirls', 'Dust Whirls Nearby', 'Dust
Whirlwinds', 'Duststorm', 'Fair', 'Fog', 'Funnel Cloud', 'Hail', 'Haze', 'Heavy ', 'Heavy Drizzle', 'Heavy Ice Pellets', 'He
avy Rain', 'Heavy Rain Shower', 'Heavy Rain Showers', 'Heavy Sleet', 'Heavy Snow', 'Heavy T-Storm', 'Heavy Thunderstorms',
'Ice Pellets', 'Light ', 'Light Drizzle', 'Light Fog', 'Light Haze', 'Light Ice Pellets', 'Light Rain', 'Light Rain Shower',
'Light Rain Showers', 'Light Sleet', 'Light Snow', 'Light Snow Shower', 'Light Snow Showers', 'Light Thunderstorms', 'Low Dr
ifting Snow', 'Mist', 'N/A Precipitation', 'Overcast', 'Partial Fog', 'Patches of Fog', 'Rain', 'Rain Shower', 'Rain Shower
s', 'Sand', 'Scattered Clouds', 'Shallow Fog', 'Showers in the Vicinity', 'Sleet', 'Small Hail', 'Smoke', 'Snow', 'Snow Grai
ns', 'Snow Nearby', 'Squalls', 'T-Storm', 'Thunder', 'Thunder in the Vicinity', 'Thunderstorm', 'Thunderstorms', 'Tornado',
'Volcanic Ash', 'Widespread Dust', 'Windy', 'Wintry Mix']
```

	Weather_Condition	Clear	Cloud	Rain	Heavy_Rain	Snow	Heavy_Snow	Fog
0	Light Rain	False	False	True	False	False	False	False
1	Light Rain	False	False	True	False	False	False	False
2	Overcast	False	True	False	False	False	False	False
3	Mostly Cloudy	False	True	False	False	False	False	False
4	Mostly Cloudy	False	True	False	False	False	False	False
...	...	...	...	...	...	...	...	...
4232536	Fair	False	False	False	False	False	False	False
4232537	Fair	False	False	False	False	False	False	False
4232538	Partly Cloudy	False	True	False	False	False	False	False
4232539	Fair	False	False	False	False	False	False	False
4232540	Fair	False	False	False	False	False	False	False

SIMPLIFICATION

Converting Boolean columns to 0 ’s and 1’s :

```
Amenity            bool
Bump               bool
Crossing           bool
Give_Way           bool
Junction           bool
No_Exit            bool
Railway            bool
Roundabout         bool
Station            bool
Stop               bool
Traffic_Calming    bool
Traffic_Signal     bool
```

```
Clear              bool
Cloud              bool
Rain               bool
Heavy_Rain         bool
Snow               bool
Heavy_Snow         bool
Fog                bool
```

Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Calming
False	False	False	False	False	False	False
False	False	False	False	False	False	False
True	False	False	False	False	False	False

Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Calming
0	1	1	0	0	1	1
0	0	1	0	0	1	0
0	0	1	0	0	1	0

TRANSFORMATION

Mapping 'Start_Time' to 'Year', 'Month', 'Weekday', 'Day' (in a year), 'Hour', and 'Minute' (in a day) :

	Start_Time
0	2016-02-08 00:37:08
1	2016-02-08 05:56:20
2	2016-02-08 06:15:39
3	2016-02-08 06:51:45
4	2016-02-08 07:53:43

Year	Month	Weekday	Day	Hour	Minute
2016	2	0	39	0	37.0
2016	2	0	39	5	356.0
2016	2	0	39	6	375.0
2016	2	0	39	6	411.0
2016	2	0	39	7	473.0

TRANSFORMATION

Replacing missing values with median for Precipitation :

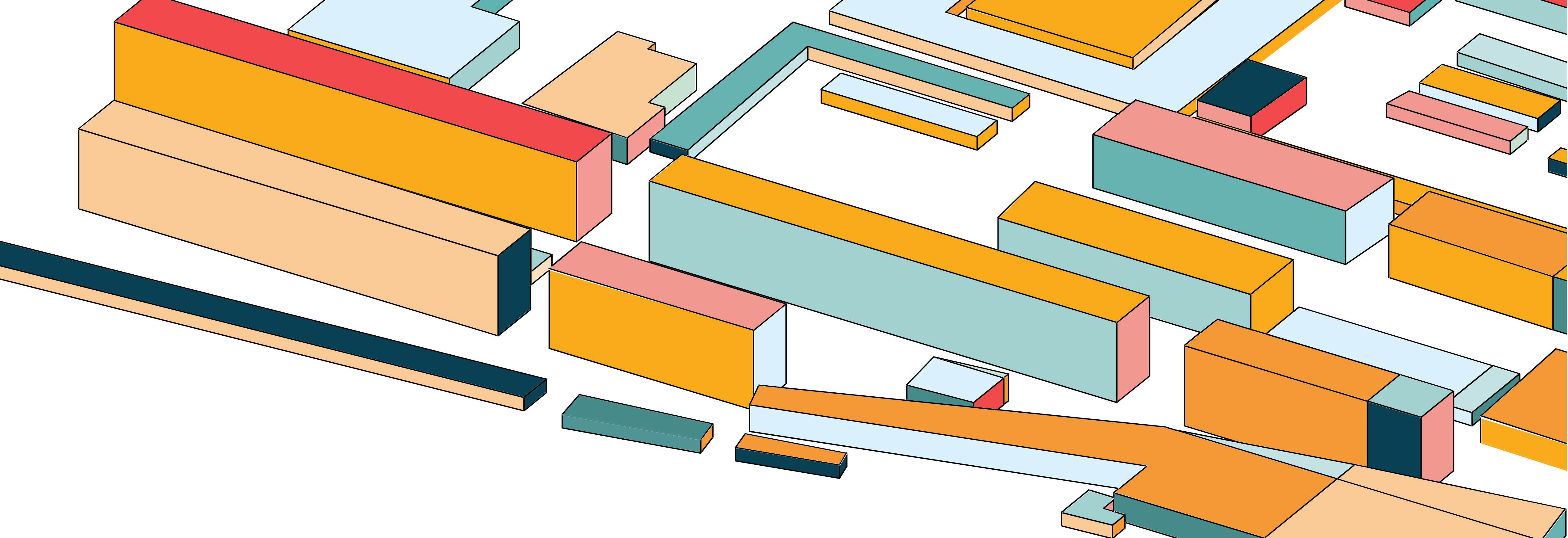
```
df['Precipitation(in)'] = df['Precipitation(in)'].fillna(df['Precipitation(in)'].median())
```

```
df['Precipitation(in)'].head(5)
```

```
0    0.00
1    0.02
2    0.02
3     NaN
4    0.01
Name: Precipitation(in), dtype: float64
```

```
df['Precipitation(in)'].head(5)
```

```
0    0.00
1    0.02
2    0.02
3    0.00
4    0.01
Name: Precipitation(in), dtype: float64
```

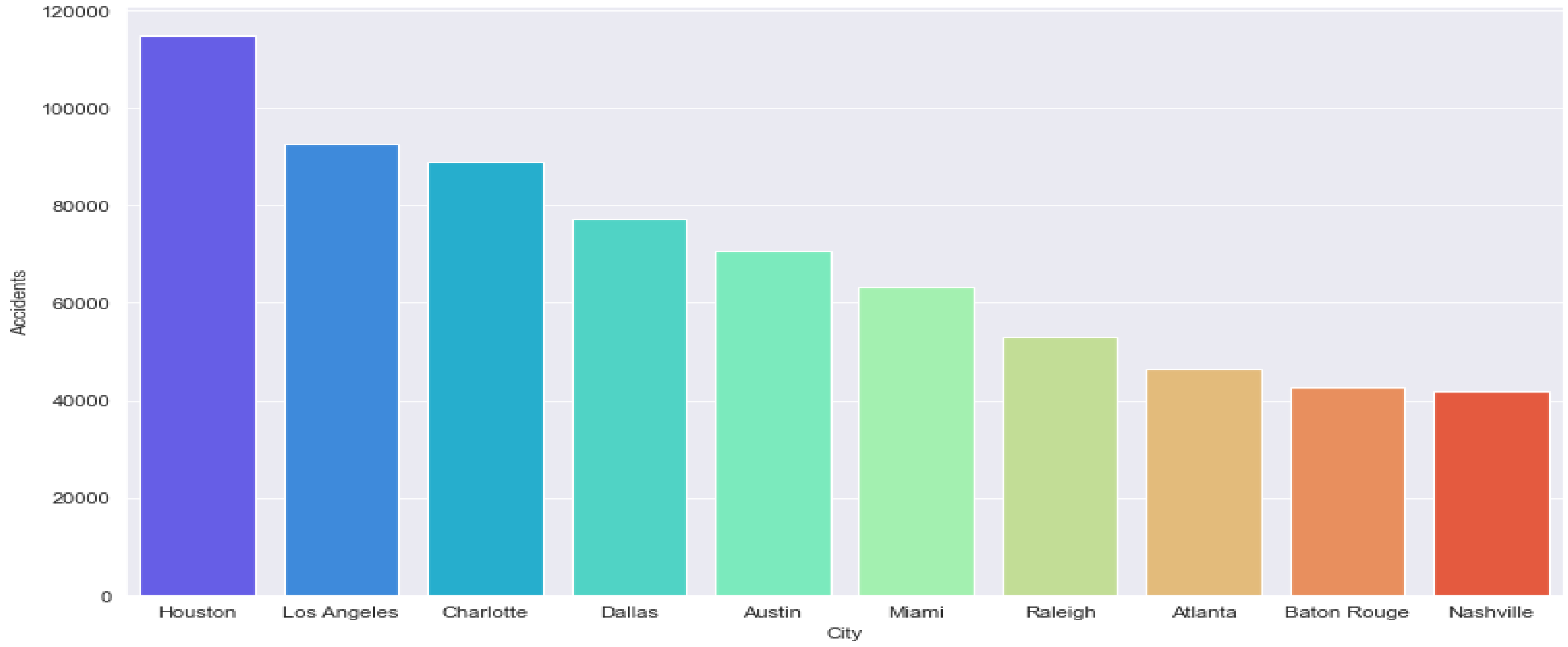


EXPLORATORY DATA ANALYSIS

1. Location Analysis
2. Time Analysis
3. Weather Conditions Analysis

LOCATION ANALYSIS

City Analysis



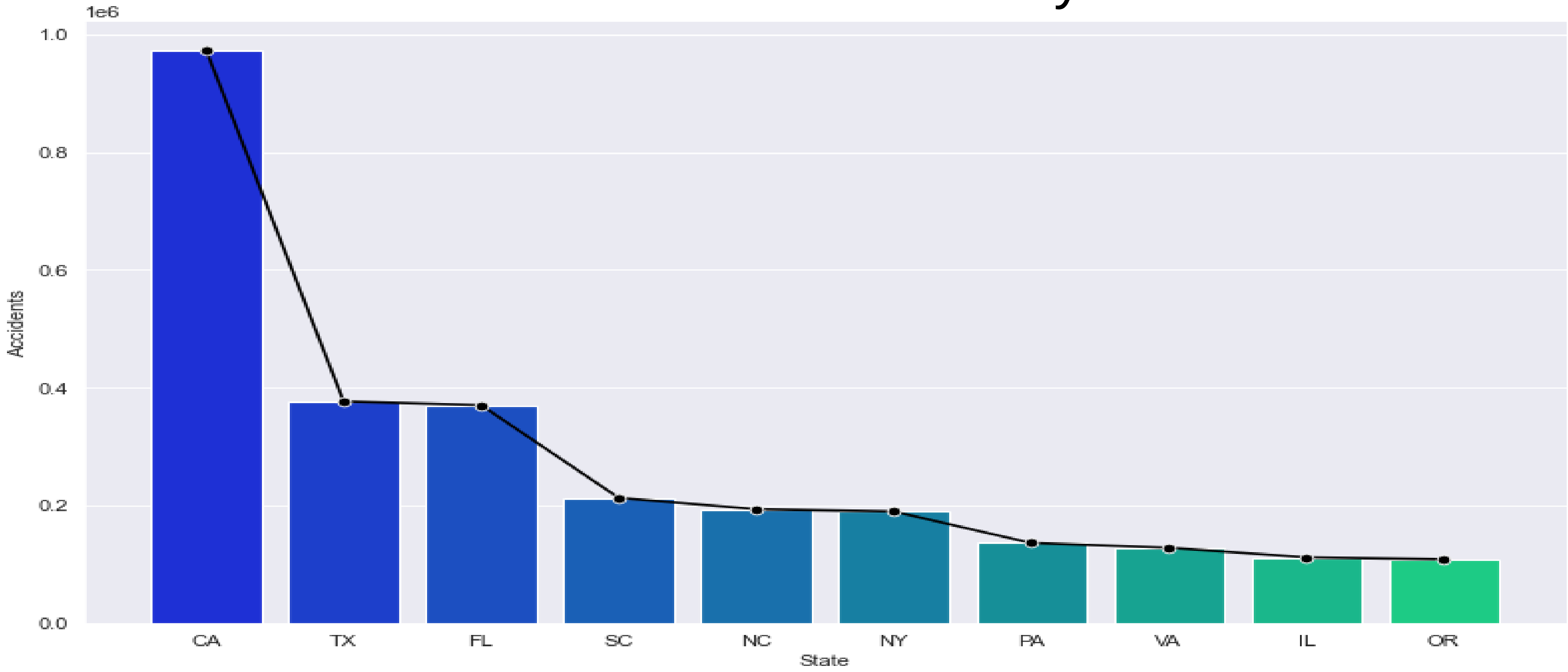
	City	Accidents
0	Houston	114905
1	Los Angeles	92701
2	Charlotte	88887
3	Dallas	77303
4	Austin	70538
5	Miami	63162
6	Raleigh	52876
7	Atlanta	46328
8	Baton Rouge	42814
9	Nashville	41850

Insights :

1. Houston is the city recording the largest number of accidents in the past 5 years in the USA, followed closely by Los Angeles and Charlotte.
2. Out of all accidents occuring in 12249 cities of the USA, 16% of them are hosted in these 10 cities.
3. 3 of the top 10 cities with most accidents are from the state of Texas.

LOCATION ANALYSIS

State Analysis



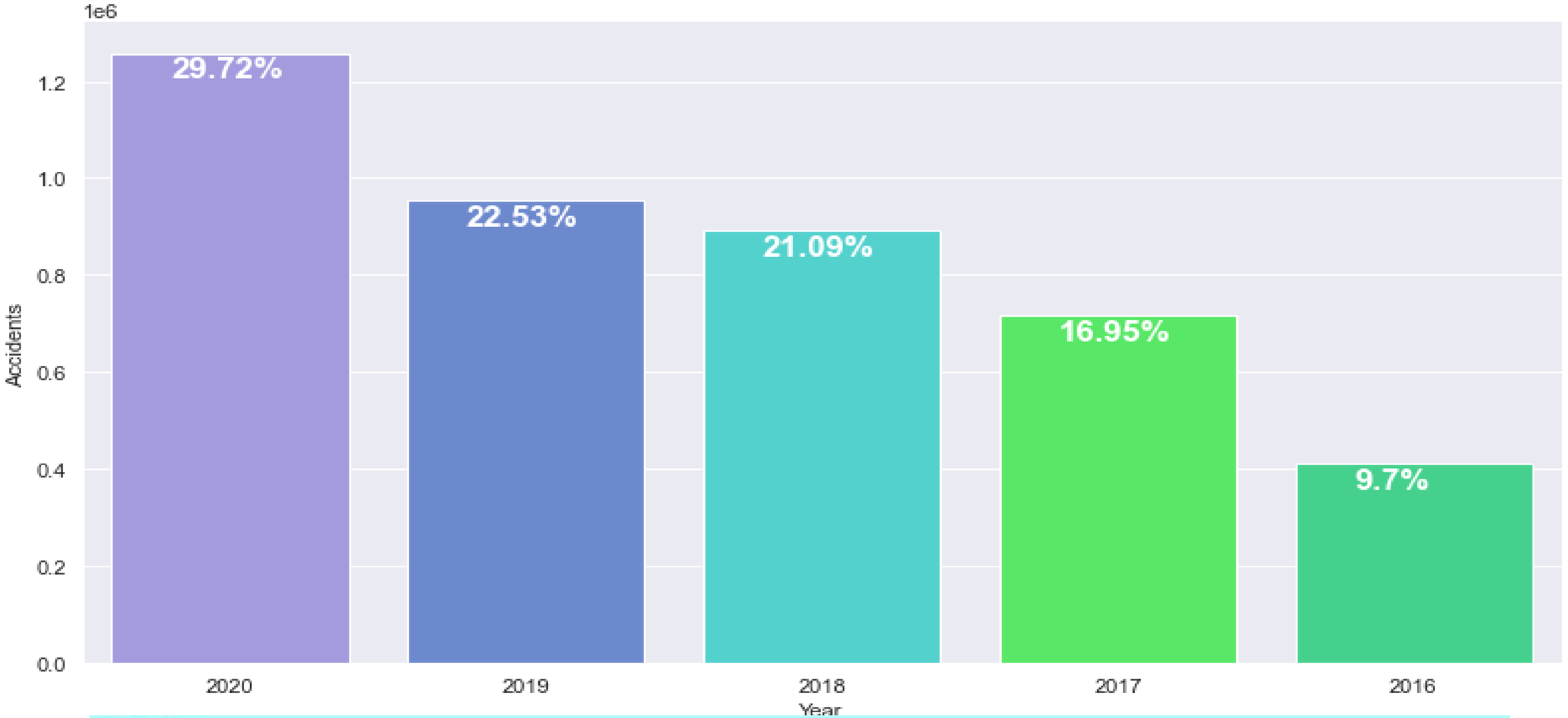
	State	Accidents
0	CA	972577
1	TX	376445
2	FL	370102
3	SC	212712
4	NC	193453
5	NY	189486
6	PA	136049
7	VA	127949
8	IL	111711
9	OR	108350
48	SD	220

Insights

1. California is the state recording most accidents,followed by Texas and Florida.
2. Out of 50 states in the USA, these 10 states make up for 66% of all recorded accidents, with the top 3 states hosting a shocking 41% of those accidents.
3. South Dakota reported the lowest number of accidents for the period 2016-2020, averaging at 44 a year.

TIME ANALYSIS

Yearly Analysis



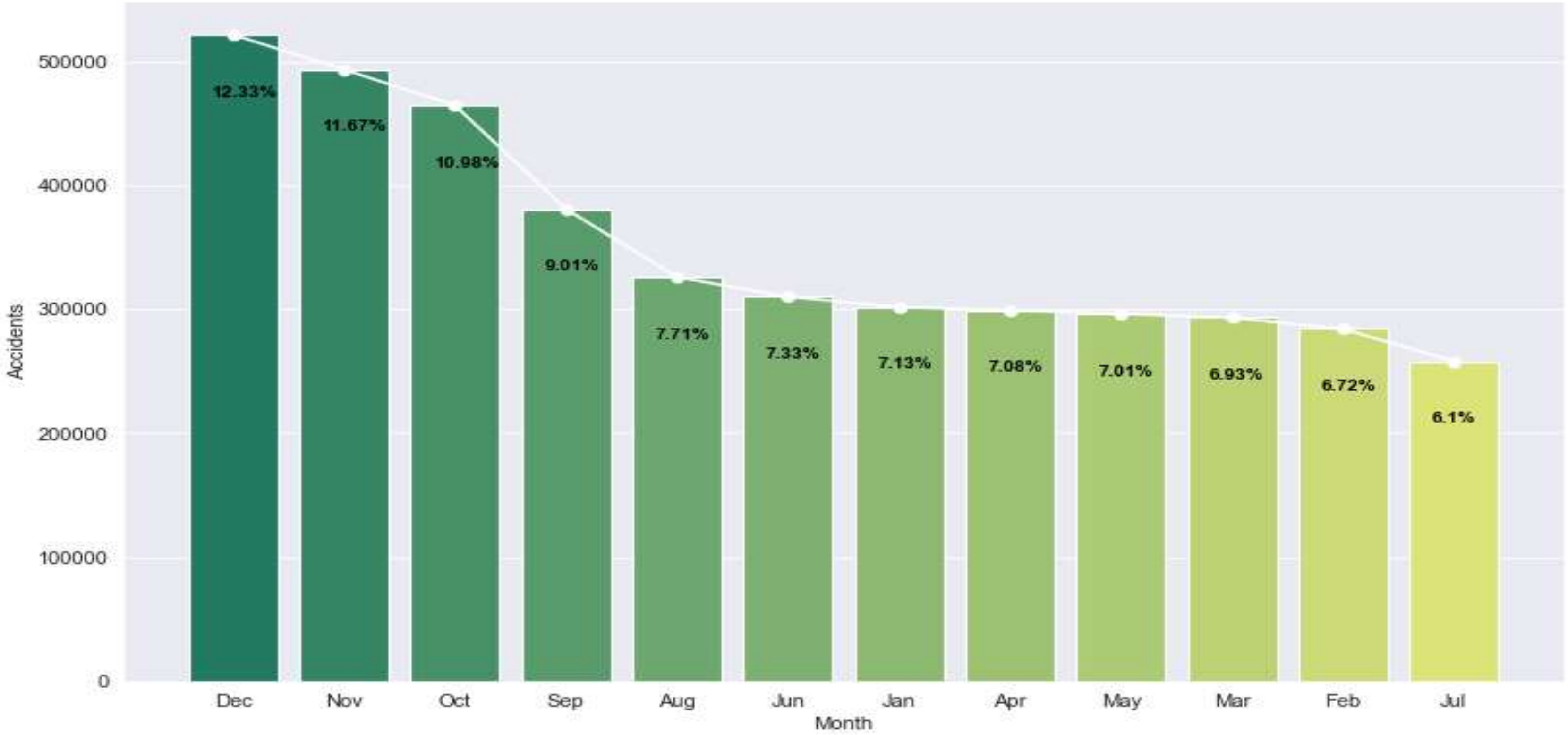
	Year	Accidents
0	2020	1258101
1	2019	953690
2	2018	892591
3	2017	717459
4	2016	410559

Insights

The number of accidents keep on increasing steadily every year to the extent where 2020 contributes to nearly 30% of all accidents recorded.

TIME ANALYSIS

Monthly Analysis



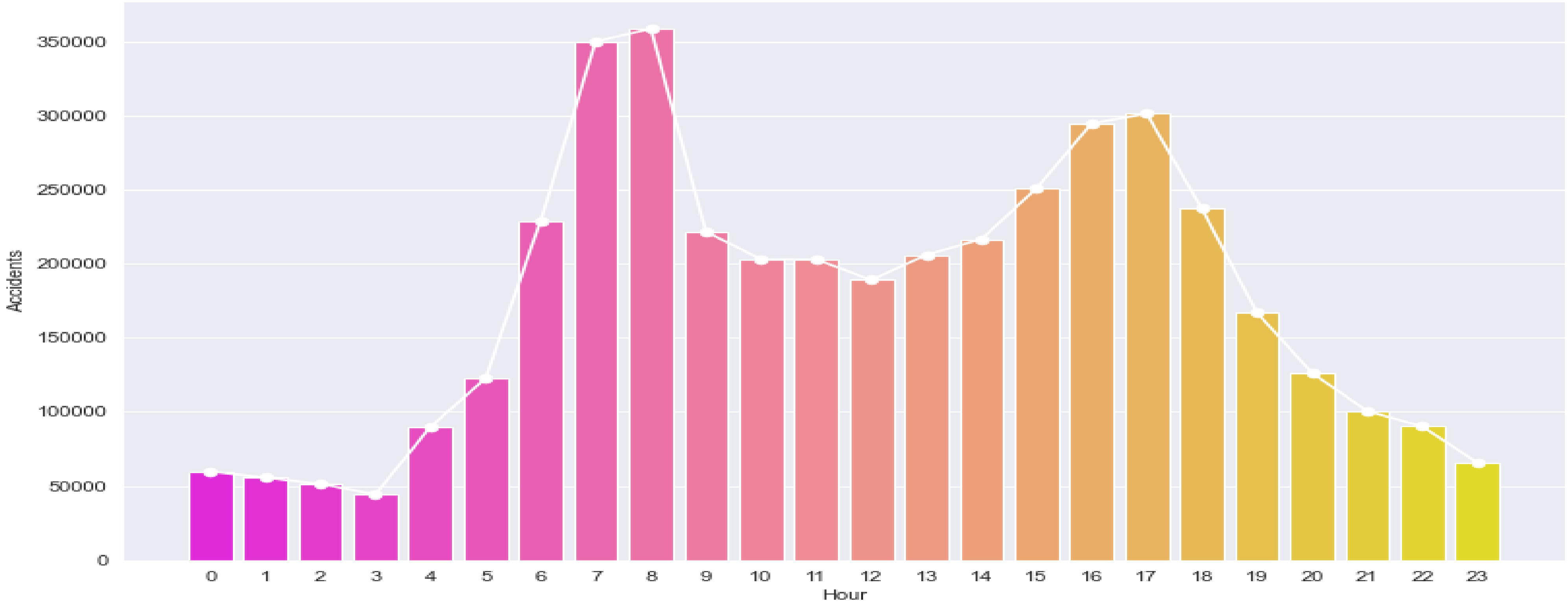
	Month	Accidents
0	Dec	521994
1	Nov	493835
2	Oct	464868
3	Sep	381189
4	Aug	326226
5	Jun	310348
6	Jan	301921
7	Apr	299477
8	May	296597
9	Mar	293371
10	Feb	284389
11	Jul	258185

Insights

For the last five years, accidents have shown a pattern of increasing towards the end of the year, where the final 3 months account for 35% of annually recorded accidents.

TIME ANALYSIS

Hourly Analysis

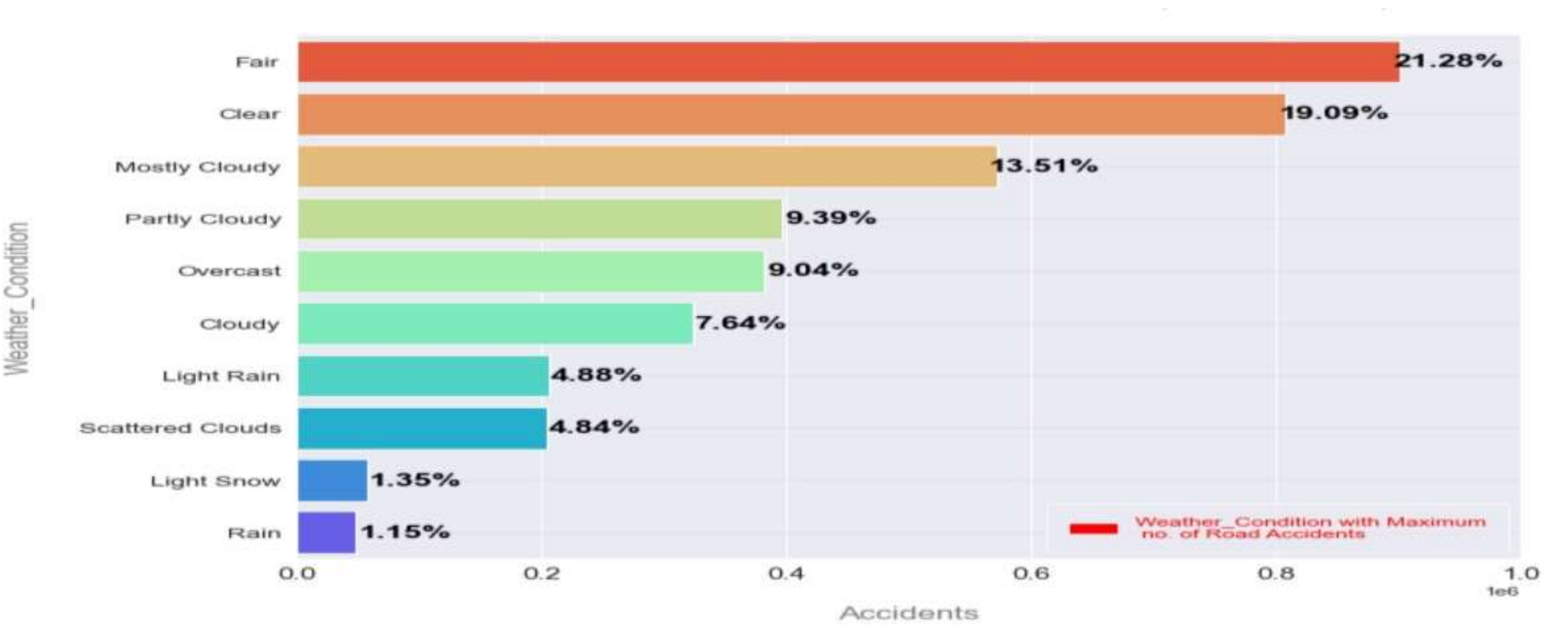


Insights

The morning times of 7-8am and evening times of 4-5pm have been the most accident prone hours, suggesting office timings might have a role to play in these accidents.

	Hour	Accidents
0	8	358419
1	7	349933
2	17	301109
3	16	294585
4	15	250797
5	18	237043
6	6	228870
7	9	221536
8	14	216502
9	13	205845
10	10	202991
11	11	202902
12	12	188986
13	19	166955
14	20	126253
15	5	122517
16	21	100517
17	22	90149
18	4	89815
19	23	65697
20	0	59772
21	1	55597
22	2	51363
23	3	44247

WEATHER CONDITIONS ANALYSIS

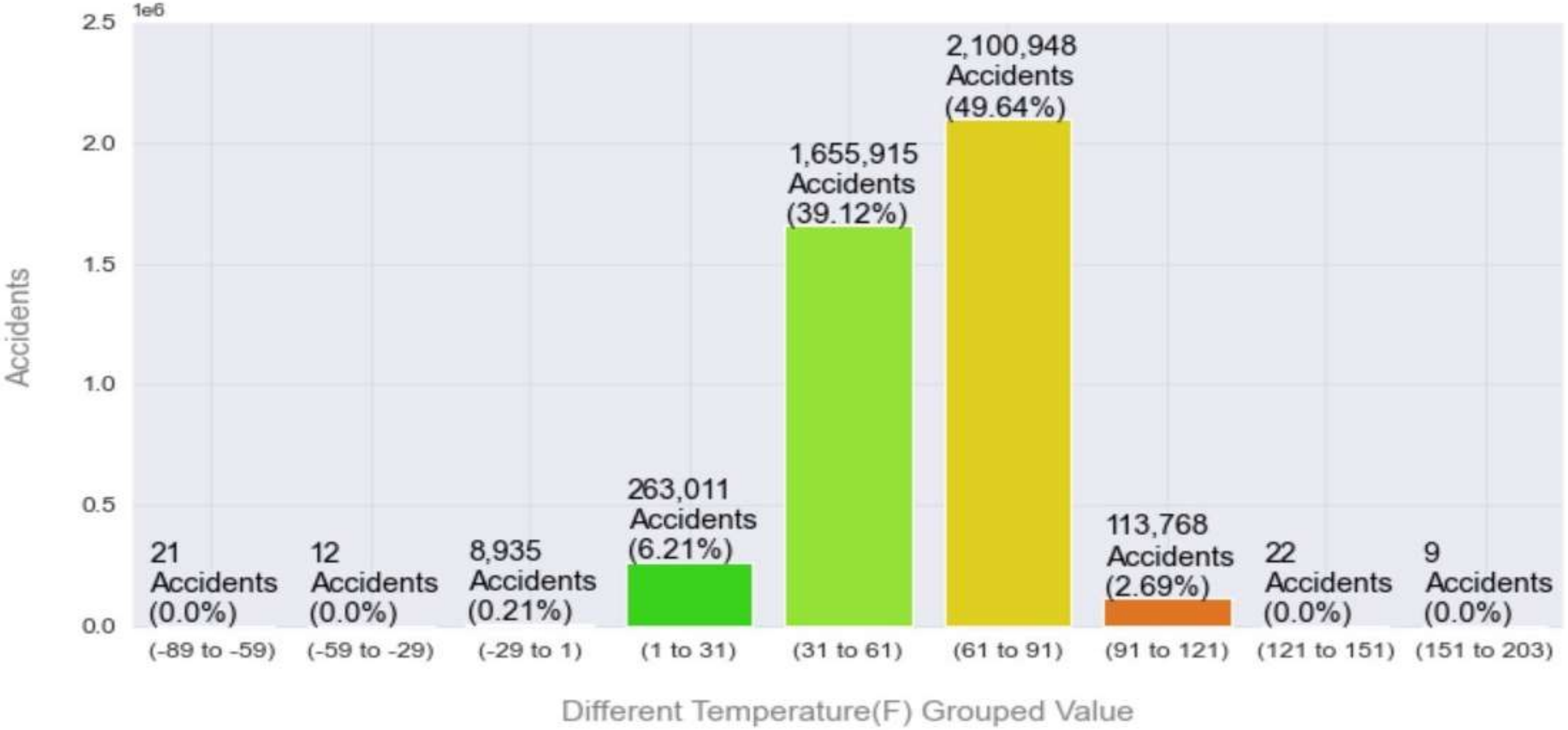


Insights

Ironically, statistics show that 40% of accidents have occured when the weather conditions are fair and clear.

WEATHER CONDITIONS ANALYSIS

Temperature Analysis



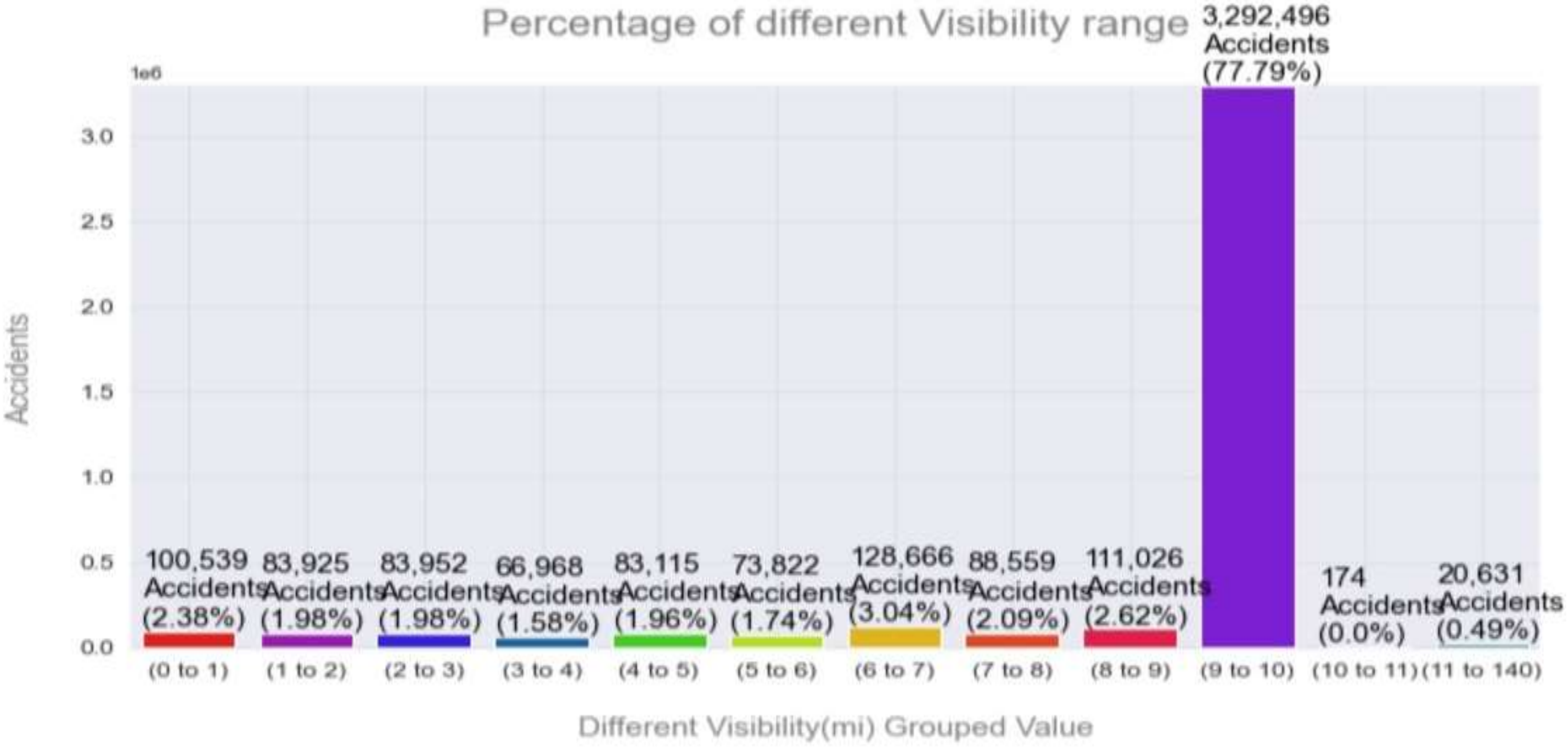
	Temperature(F)	Accidents
0	68.0	91544
1	77.0	90039
2	59.0	86344
3	73.0	84210
4	63.0	80070
...	...	...
835	111.4	1
836	119.0	1
837	113.9	1
838	113.4	1
839	-32.8	1

Insights

The temperature range of 61(F)-91(F) accounts for nearly 50% of all accidents recorded.

WEATHER CONDITIONS ANALYSIS

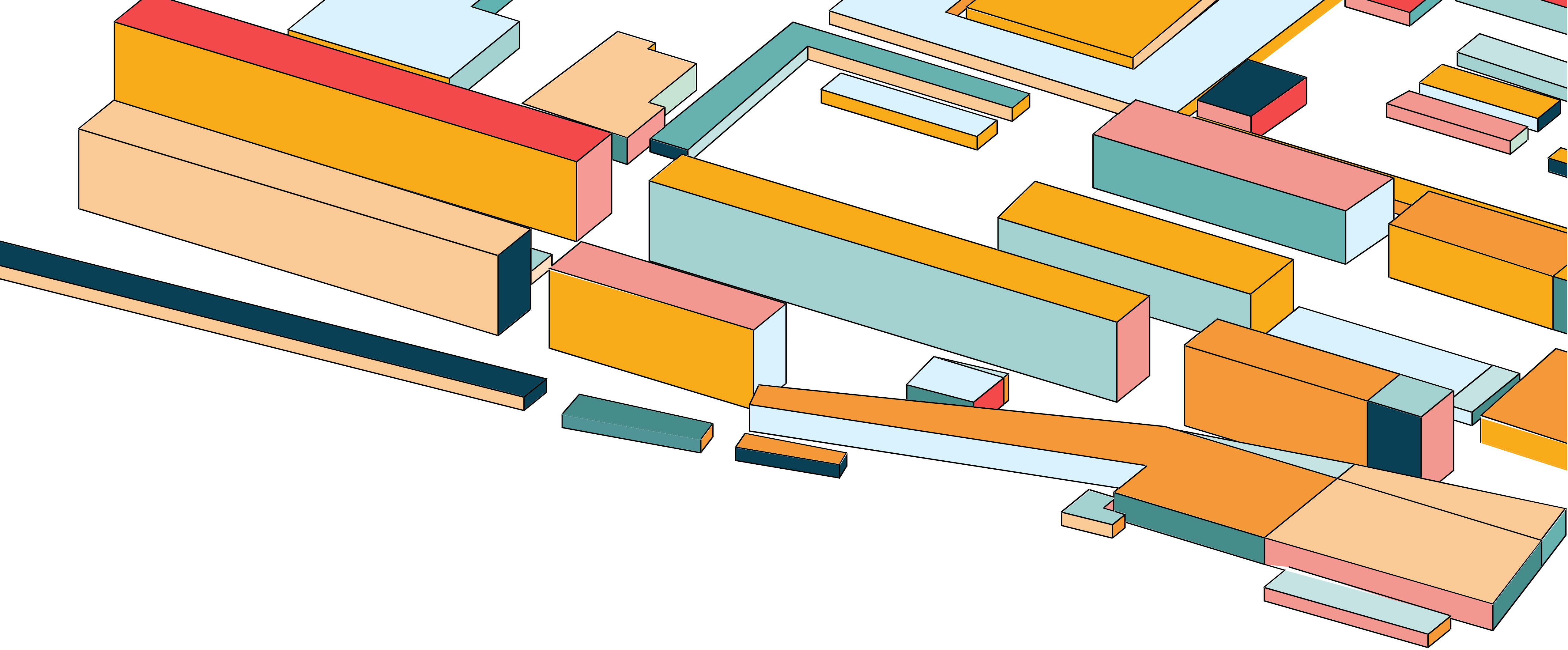
Visibility Analysis



Insights

77% of accidents occured when drivers had visibility between 9-10 miles.

	Visibility(mi)	Accidents
0	10.00	3292390
1	7.00	128662
2	9.00	111023
3	8.00	88557
4	5.00	83097
...	...	...
82	101.00	1
83	16.00	1
84	0.31	1
85	3.20	1
86	43.00	1



MACHINE LEARNING

CLASS IMBALANCE

We have a class imbalance issue because there are 2440379 instances of accidents with severity 1 but not as many instances with severity 0, 2 & 3.

```
In [36]: y = df['Severity'].copy()
         X = df.drop('Severity', axis=1).copy()

In [37]: y.unique()

Out[37]: array([3, 2, 4, 1], dtype=int64)

In [38]: y = y-1

In [39]: pd.DataFrame(y).value_counts()

Out[39]: Severity
         0      25536
         1    2440379
         2     149543
         3     125058
         Name: Severity, dtype: int64
```

UNDER-SAMPLING

We use under-sampling to reduce the number of instances of accidents with severity 1 from 2440379 down to 200000.

UnderSampling

In [40]:
```
from imblearn.under_sampling import NearMiss

sampling_strategy = {1: 200000}
nr = NearMiss(sampling_strategy=sampling_strategy)

X, y = nr.fit_resample(X, y.ravel())
```

In [41]:
```
pd.DataFrame(y).value_counts()
```

Out[41]:
```
1    200000
2    149543
3    125058
0     25536
dtype: int64
```

OVERSAMPLING

We use oversampling to increase the number of instances of accidents with severity 0, 2 & 3 to 200000.

OverSampling

```
In [42]: from imblearn.over_sampling import RandomOverSampler

         sampling_strategy1 = {0: 200000, 2: 200000, 3: 200000}
         oversample = RandomOverSampler(sampling_strategy=sampling_strategy1)

         X_miss, y_miss = oversample.fit_resample(X, y)

In [43]: pd.DataFrame(y_miss).value_counts()

Out[43]: 0    200000
         1    200000
         2    200000
         3    200000
         dtype: int64
```

TRAIN TEST SPLIT

We have 640000 examples in our training dataset and 160000 examples in out testing dataset.

Train Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_miss, y_miss, train_size=0.8, random_state=100)
```

```
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(640000, 100) (640000,) (160000, 100) (160000,)
```

STANDARDIZATION

Our dataset contains variables that are different in scale – **Temperature** can have values on a scale of **32-212** and **Precipitation** can have values on scale of **0-2(inches)**.
As these two columns are different in scale, they are Standardized to have common scale while building machine learning model.

Standardization

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

XGBOOST

Xgboost

```
model = xgb.XGBClassifier()
model.fit(X_train, y_train)
```

```
▸ XGBClassifier
```

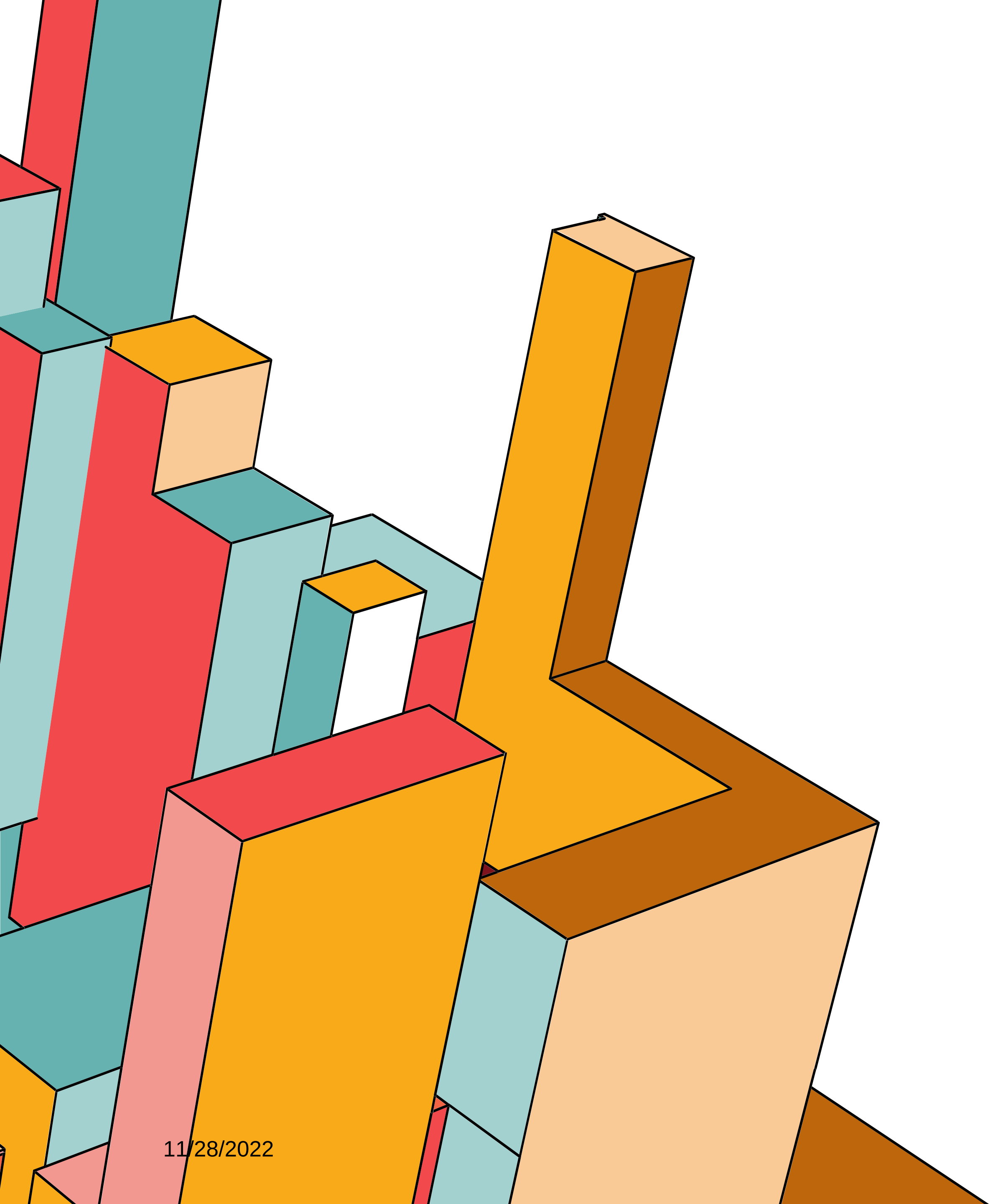
```
predicted_y = model.predict(X_test)
```

```
from sklearn import metrics
print(metrics.classification_report(y_test, predicted_y))
```

```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     40040
           1       0.85      0.81      0.83     39971
           2       0.74      0.75      0.74     39796
           3       0.82      0.74      0.78     40193

    accuracy                           0.82    160000
   macro avg       0.81      0.82      0.81    160000
weighted avg       0.81      0.82      0.81    160000
```



DECISION TREE

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
```

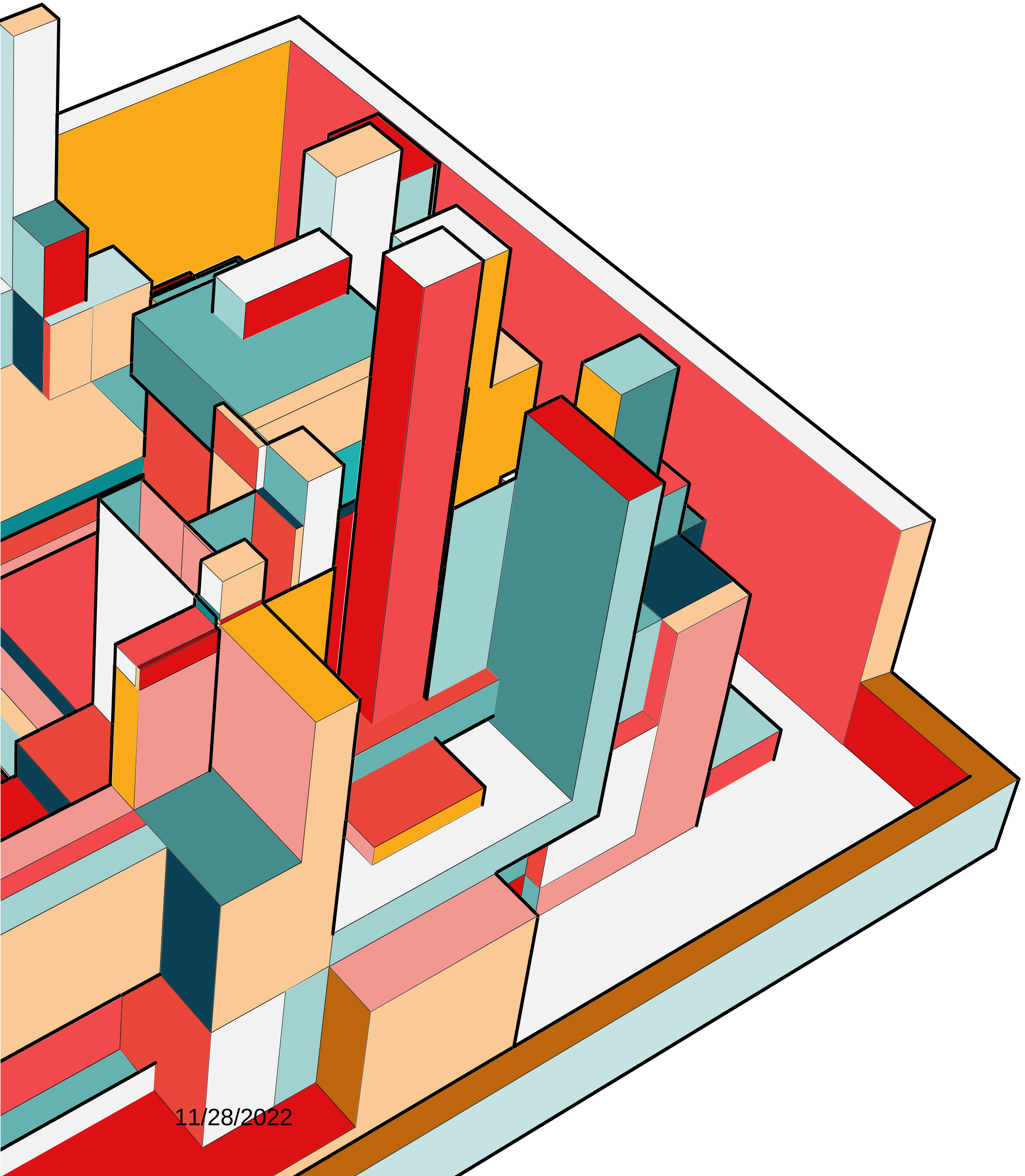
```
y_pred = clf.predict(X_test)
```

```
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     40040
           1       0.92      0.85      0.88     39971
           2       0.84      0.83      0.83     39796
           3       0.85      0.88      0.87     40193

    accuracy                           0.89    160000
   macro avg       0.89      0.89      0.89    160000
weighted avg       0.89      0.89      0.89    160000
```



FLASK APP

Please go the following link to predict the severity of an accident on a scale of 4 –

https://accident-severity.herokuapp.com/

THANK YOU