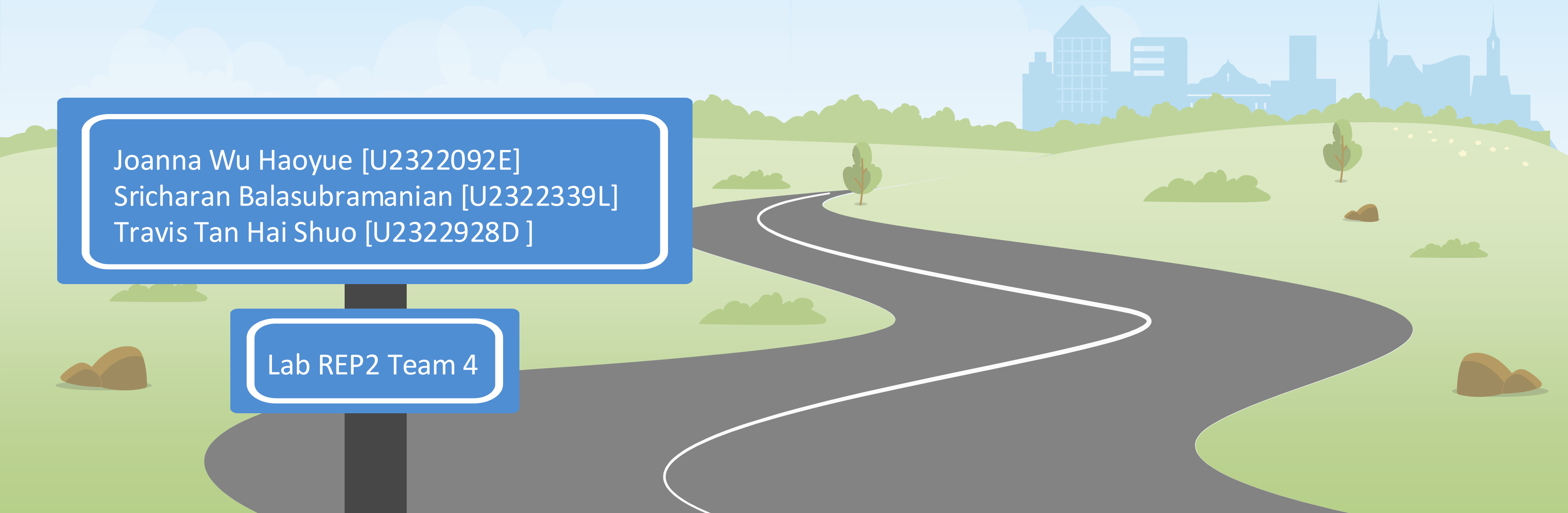
SC1015 Intro to DSAI:

Enhancing road safety in California with data-driven insights

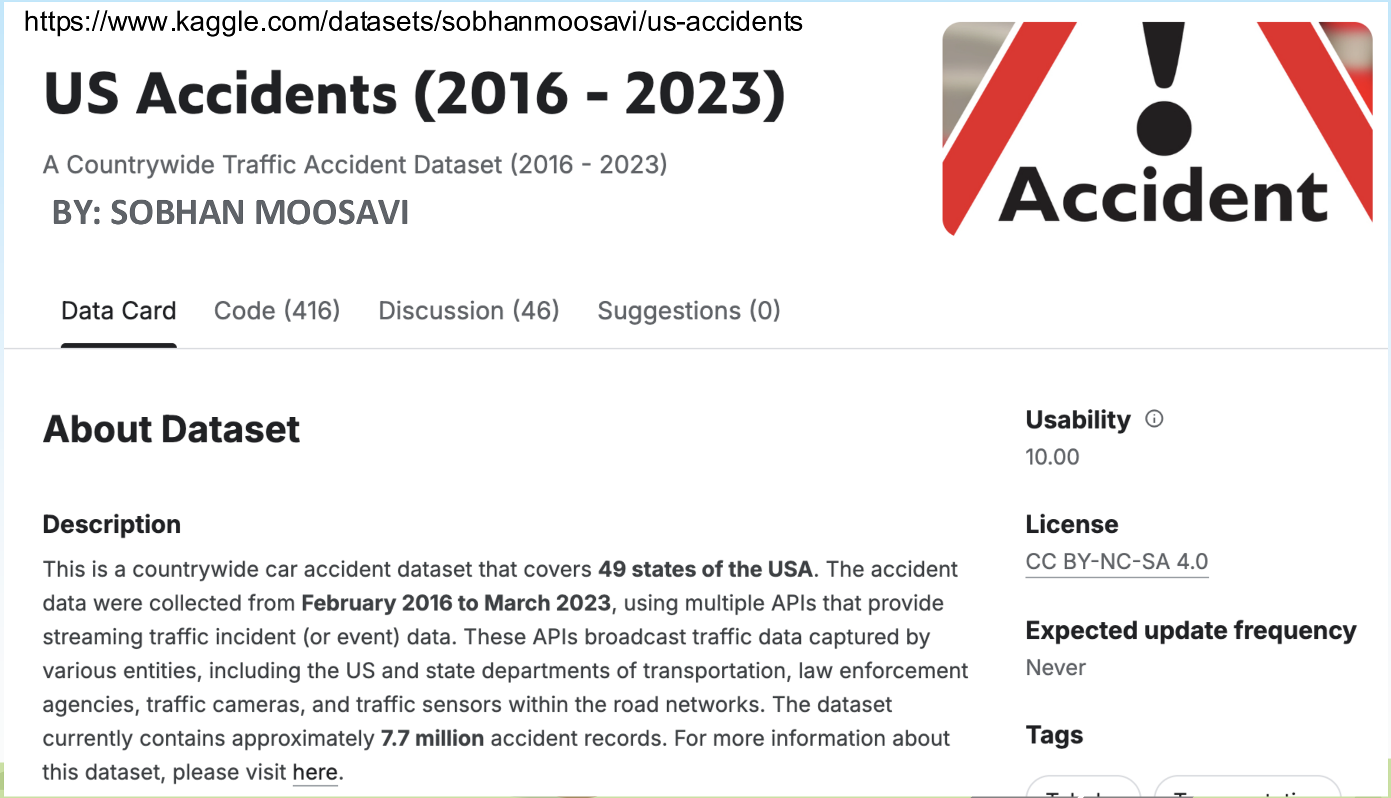


Joanna Wu Haoyue [U2322092E]
Sricharan Balasubramanian [U2322339L]
Travis Tan Hai Shuo [U2322928D]

Lab REP2 Team 4

Chosen Dataset

https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents



US Accidents (2016 - 2023)

A Countrywide Traffic Accident Dataset (2016 - 2023)

BY: SOBHAN MOOSAVI

Data Card Code (416) Discussion (46) Suggestions (0)

About Dataset

Description

This is a countrywide car accident dataset that covers **49 states of the USA**. The accident data were collected from **February 2016 to March 2023**, using multiple APIs that provide streaming traffic incident (or event) data. These APIs broadcast traffic data captured by various entities, including the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road networks. The dataset currently contains approximately **7.7 million** accident records. For more information about this dataset, please visit here.

Usability
10.00

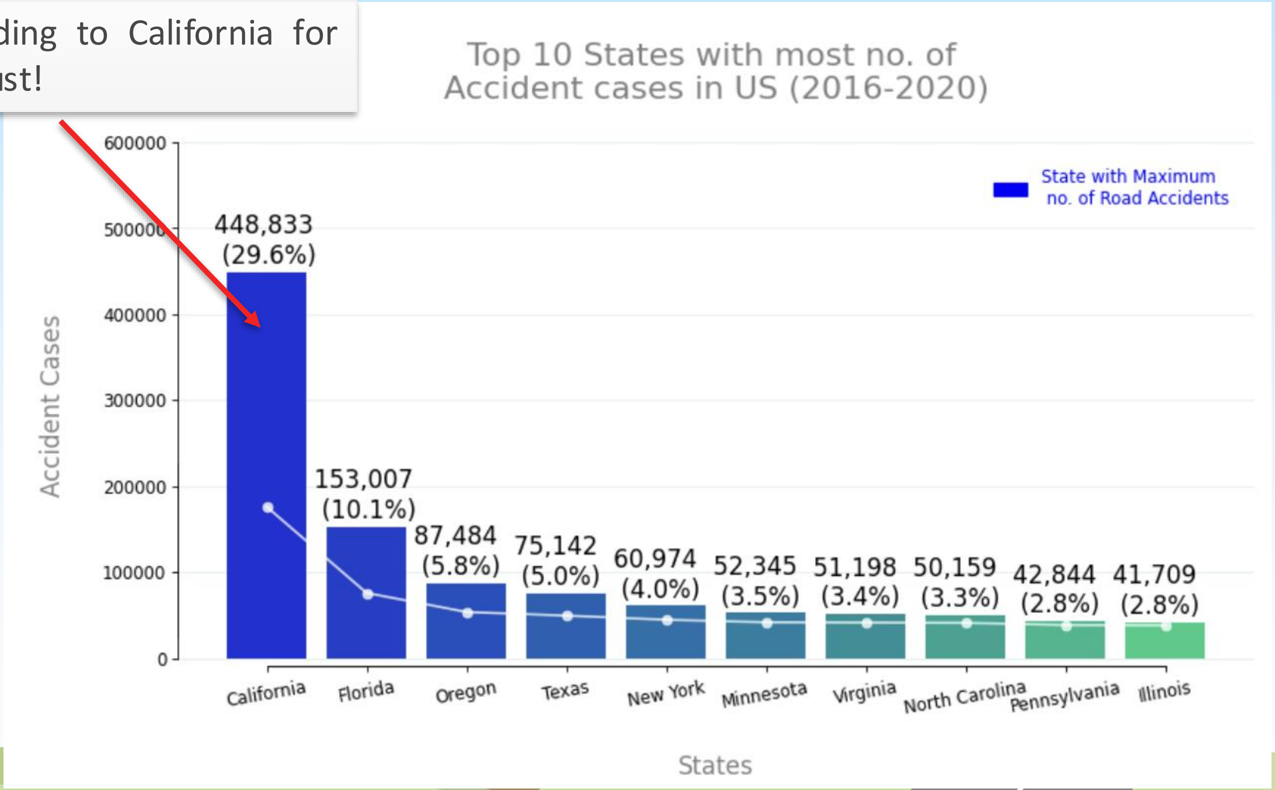
License
CC BY-NC-SA 4.0

Expected update frequency
Never

Tags

Chosen Dataset

We will be heading to California for exchange in August!



Problem Definition

Analyse the **times of day/night** and **weather conditions** where Californian **cities are most dangerous to drive in.**

1. This helps emergency services **allocate resources optimally, reducing fatality of road accidents.**
2. This can be integrated into GPS to **caution drivers** when they are driving in accident-prone conditions, **reducing frequency of road accidents.**

Data Preparation & Cleaning

Data Type Conversion

```
Street            object
City              object
County            object
State             object
Zipcode           object
```

```
Street          category
City            category
County          category
State           category
Zipcode         category
```

Dataset’s features were **converted into appropriate types**, making data handling easier for ML models.

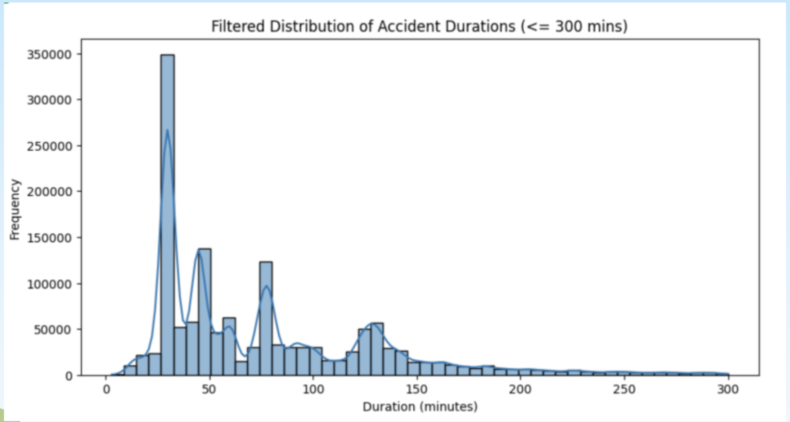
```
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
df['End_Time'] = pd.to_datetime(df['End_Time'], errors='coerce')
```

Time elements were **parsed** (time columns were converted into datetime objects).

Date and time were separated. Only **time feature was extracted** as date was meaningless in the problem context.

Feature Engineering

Feature Creation



```
df['Duration'] = df['End_Time'] - df['Start_Time']
```

Created a new feature 'Duration' to visualise duration of accidents, and to help decide if 'End_Time' is relevant to our analysis

Feature Selection

Number of unique...

```
State                1
Zipcode         129022
Country              1
Timezone             2
```

We dropped features that were irrelevant to our problem statement. E.g. 'Country', 'State' etc. are the same for each data point, and thus can be dropped. Twilights are alternative ways to measure time. 'End_Time' is shown to be irrelevant

Handling Missing Data

Critical features with low NULL %

These rows were **dropped** as the change in results is **negligible**, but quality of dataset will be **greatly improved.**

Weather features

These cells were filled with **median values** to reflect the central tendency of weather conditions

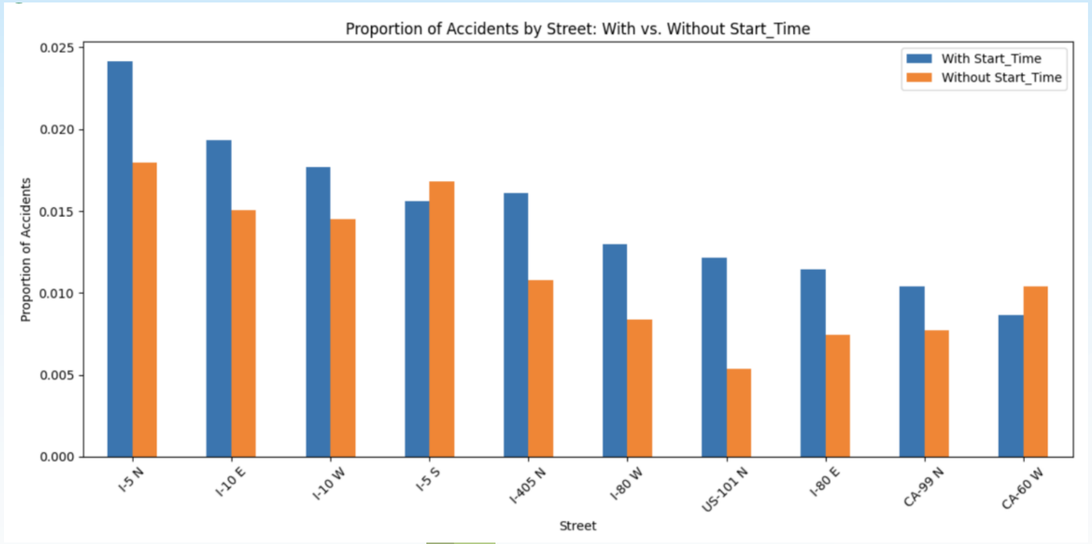
High NULL %

Some features (Wind_Chill) were dropped as it is **strongly correlated** with other features like Temperature. **Precipitation NULLs are inputted with 0**.

Handling Missing Data

'Start_Time' → 10% of values are NULL!

To decide how to handle NULL accident time values, we compared the **proportions** of the most accident-prone streets that have a recorded time and those that do not. Ultimately, **no major disproportion** was observed, implying the results would not be heavily skewed with the removal of these NULL points.



Outlier Handling

Data Normalisation

Noticing a wide spread of numerical data (many unique values), we adjusted the data so the values fall within a **specific range**

Outlier Removal

Outliers were detected in various numerical categories upon observing the spread of data in boxplots. They were removed using the **Interquartile** method.

Exploratory Data Analysis

Regions

Are trends **homogenous** all across California, or are there **specific places** to consider?

Day and Time

What trends can we identify in the **frequency** of accidents against the **time of the day**, and the **day of the week**?

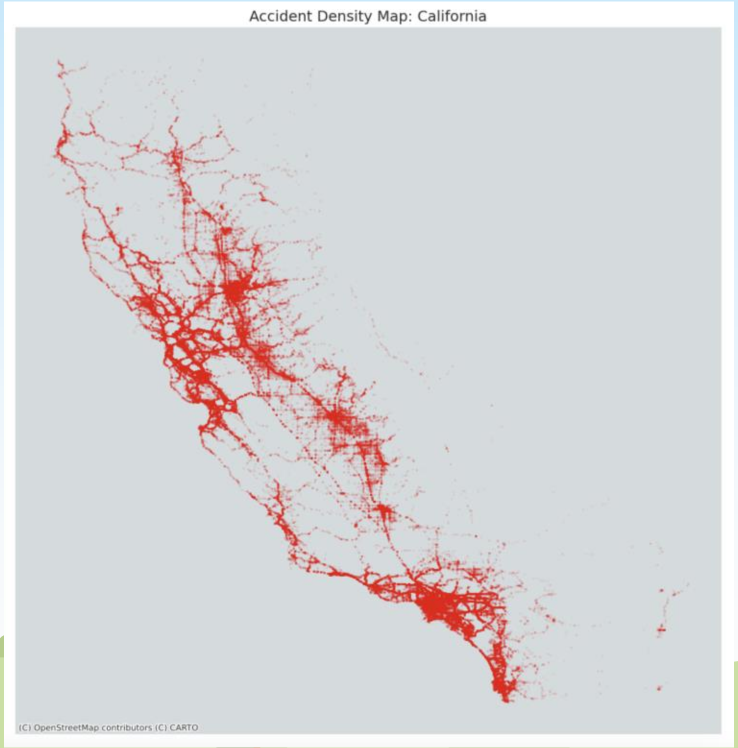
Weather Conditions

In the regions, do accidents occur in all weather conditions or only in **certain weather conditions**?

General Regional Analysis

We see that the distribution is centred in some specific parts of California, rather than being distributed evenly all around the state.

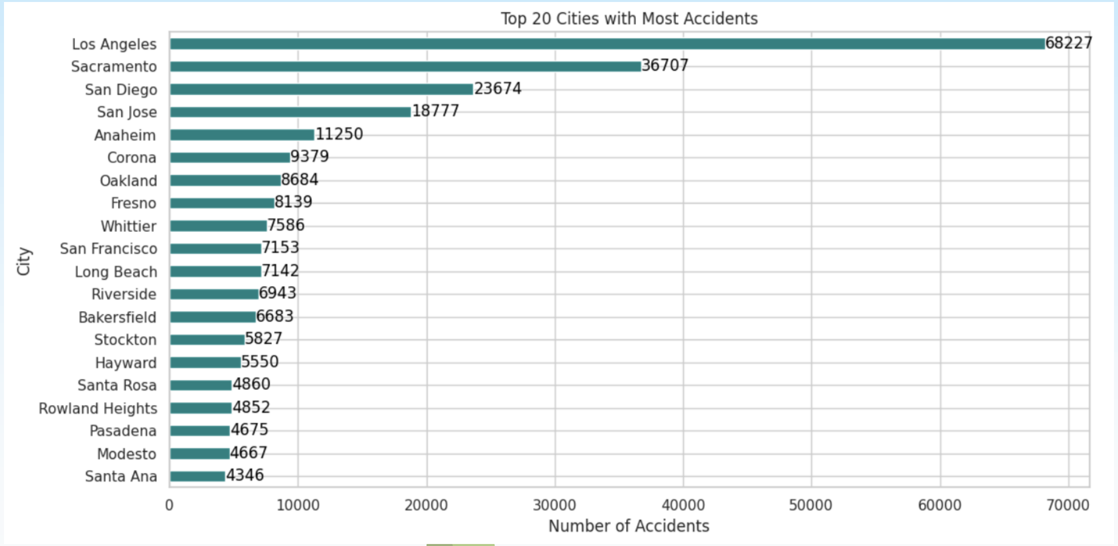
Narrows scope of our discussion and analysis


Accident Density Map: California
(C) OpenStreetMap contributors (C) CARTO

City Analysis

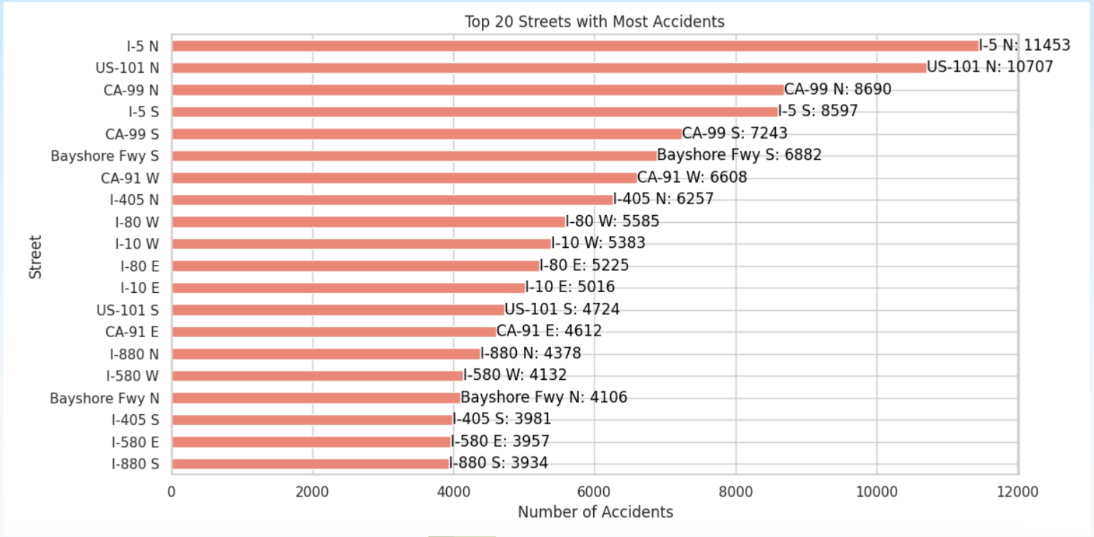
We see a clear spike in some cities, and this validates our approach of looking at cities within California, rather than treating the entire state as one big homogenous space

Care must be taken within cities, and this is a foundation for our future model



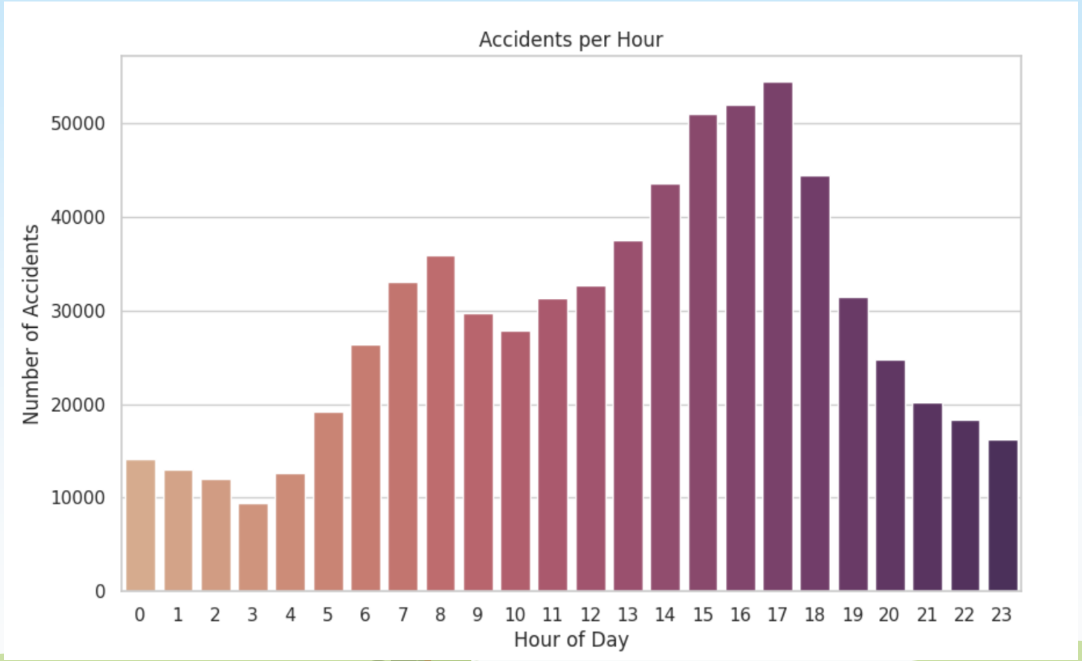
Street Analysis

The streets show a more homogenous mix, but the streets remain meaningful for us to look at due to their direct relevance to our use case. However, the cardinality is a weakness that makes it hard to be analysed



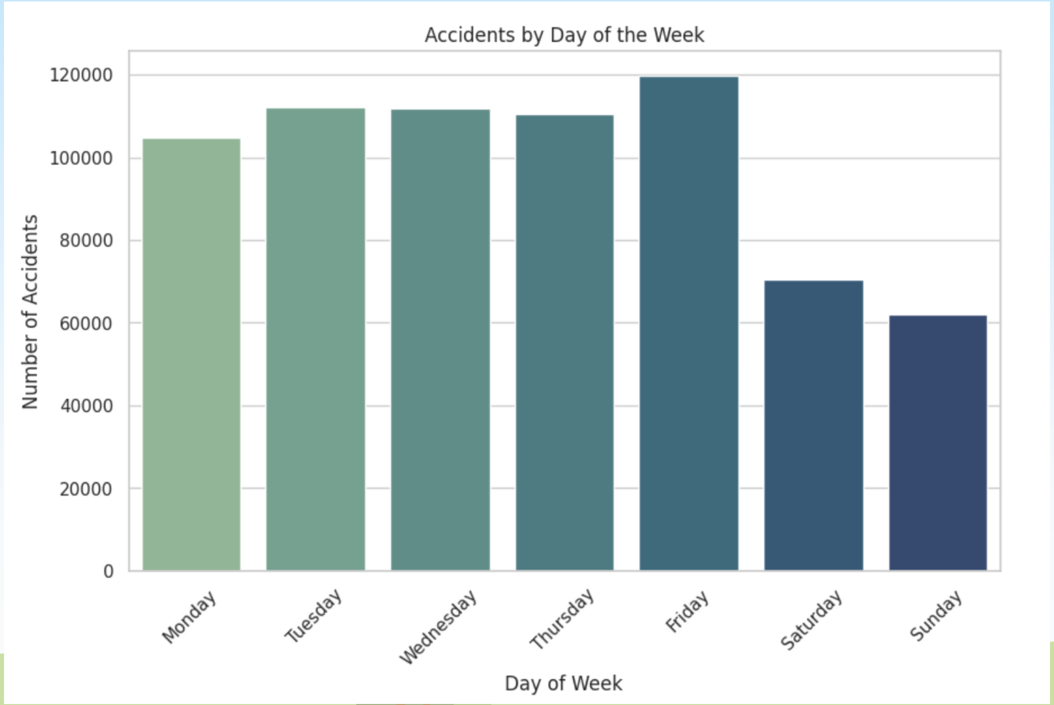
Time Analysis

We see a local spike between 7AM and 9AM, and a global spike between 4PM and 7PM. These are peak hours, and clearly this is another dimension for our analysis



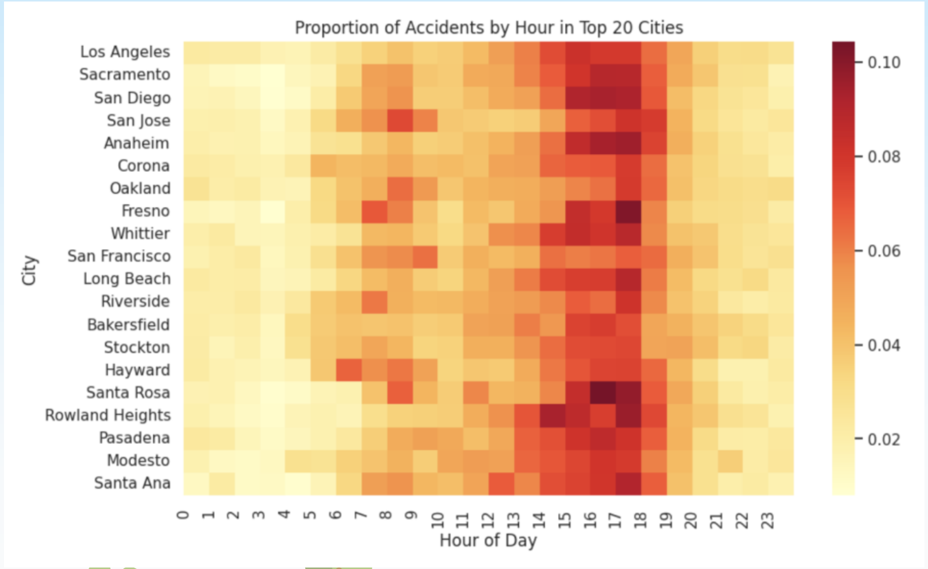
Day Analysis

A clear distinction between weekdays and weekends. A specific spike on Fridays, possibly an indicator of more haphazard driving



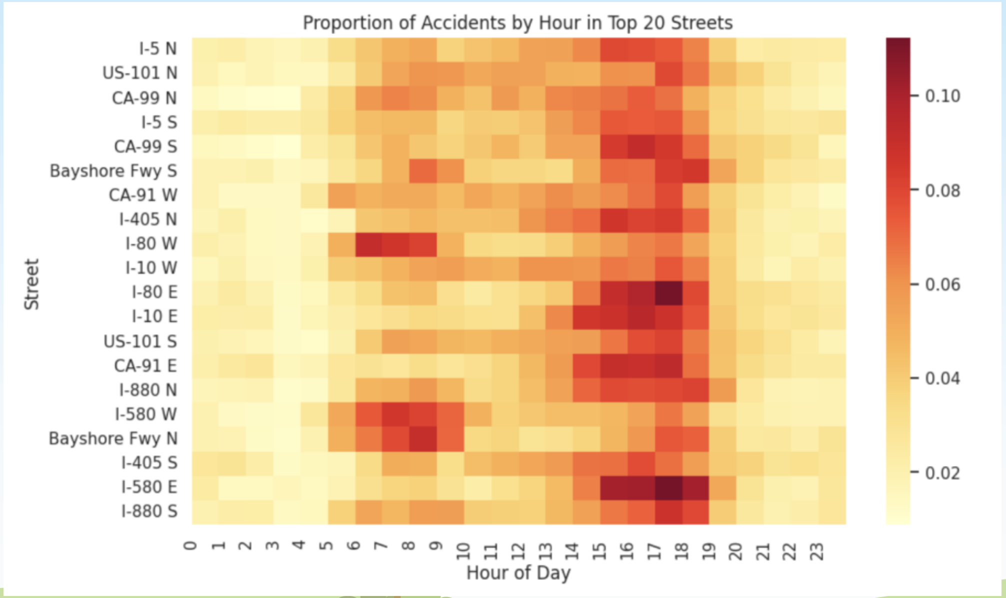
Cross Validation - City

2 clear spikes seen across all cities during the peak hours in the morning and evening



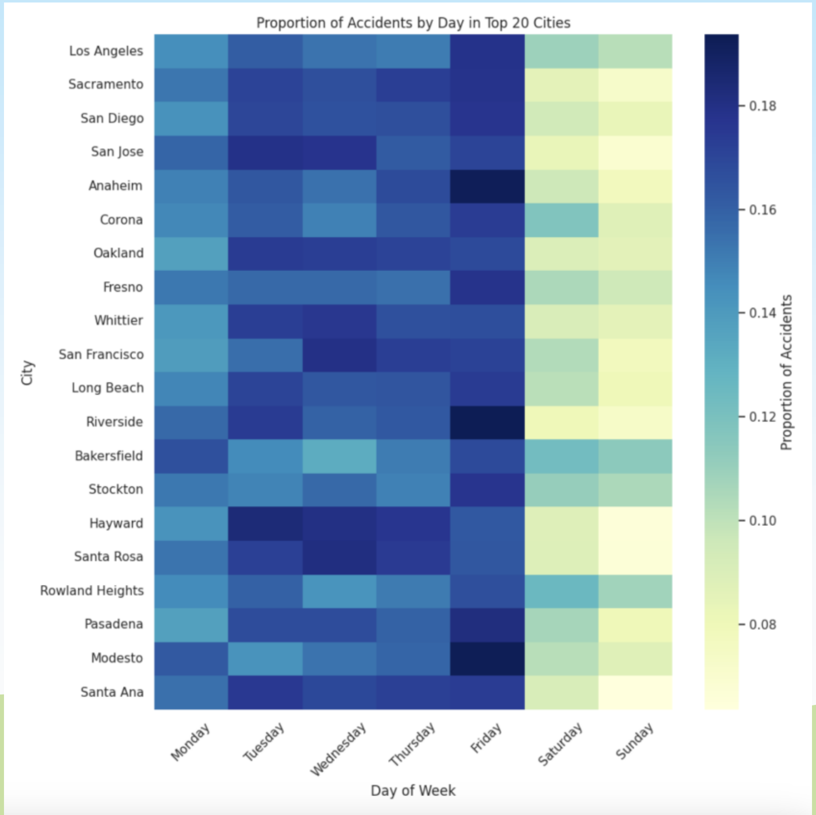
Cross Validation - Street

Sets the foundation for time-series and classification predictions based on time across cities



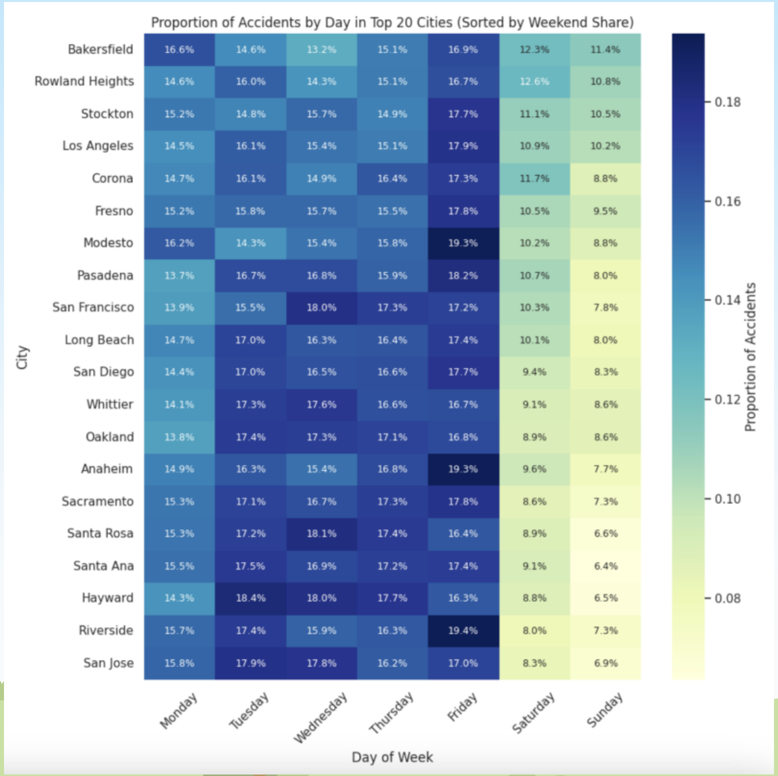
Cross Validation - City

The day trends are relevant to the cities with largest number of accidents, validating our focus on it



Cross Validation - City

There are no major outliers to this day-based trend. So, our analysis can be used to explain accident trends across the state without concerns of over-generalisation of trends without a regional focus.

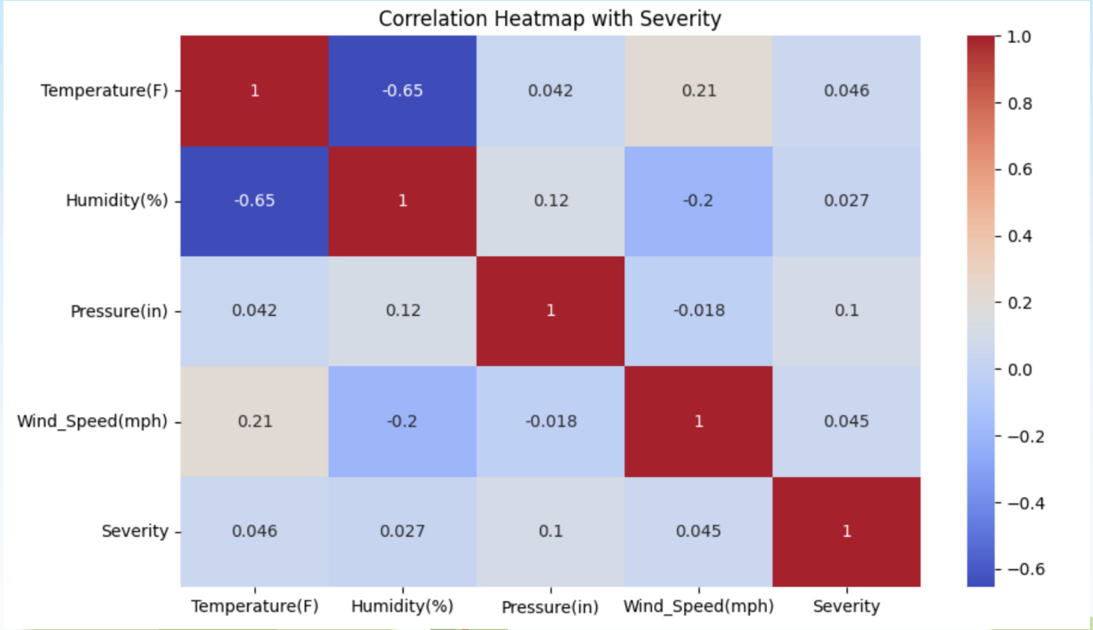


City	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Bakersfield	16.6%	14.6%	13.2%	15.1%	16.9%	12.3%	11.4%
Rowland Heights	14.6%	16.0%	14.3%	15.1%	16.7%	12.6%	10.8%
Stockton	15.2%	14.8%	15.7%	14.9%	17.7%	11.1%	10.5%
Los Angeles	14.5%	16.1%	15.4%	15.1%	17.9%	10.9%	10.2%
Corona	14.7%	16.1%	14.9%	16.4%	17.3%	11.7%	8.8%
Fresno	15.2%	15.8%	15.7%	15.5%	17.8%	10.5%	9.5%
Modesto	16.2%	14.3%	15.4%	15.8%	19.3%	10.2%	8.8%
Pasadena	13.7%	16.7%	16.8%	15.9%	18.2%	10.7%	8.0%
San Francisco	13.9%	15.5%	18.0%	17.3%	17.2%	10.3%	7.8%
Long Beach	14.7%	17.0%	16.3%	16.4%	17.4%	10.1%	8.0%
San Diego	14.4%	17.0%	16.5%	16.6%	17.7%	9.4%	8.3%
Whittier	14.1%	17.3%	17.6%	16.6%	16.7%	9.1%	8.6%
Oakland	13.8%	17.4%	17.3%	17.1%	16.8%	8.9%	8.6%
Anaheim	14.9%	16.3%	15.4%	16.8%	19.3%	9.6%	7.7%
Sacramento	15.3%	17.1%	16.7%	17.3%	17.8%	8.6%	7.3%
Santa Rosa	15.3%	17.2%	18.1%	17.4%	16.4%	8.9%	6.6%
Santa Ana	15.5%	17.5%	16.9%	17.2%	17.4%	9.1%	6.4%
Hayward	14.3%	18.4%	18.0%	17.7%	16.3%	8.8%	6.5%
Riverside	15.7%	17.4%	15.9%	16.3%	19.4%	8.0%	7.3%
San Jose	15.8%	17.9%	17.8%	16.2%	17.0%	8.3%	6.9%

Weather Analysis

Unable to remove any dimensions for weather metrics, due to low cross-correlations. All these factors remain in the model.

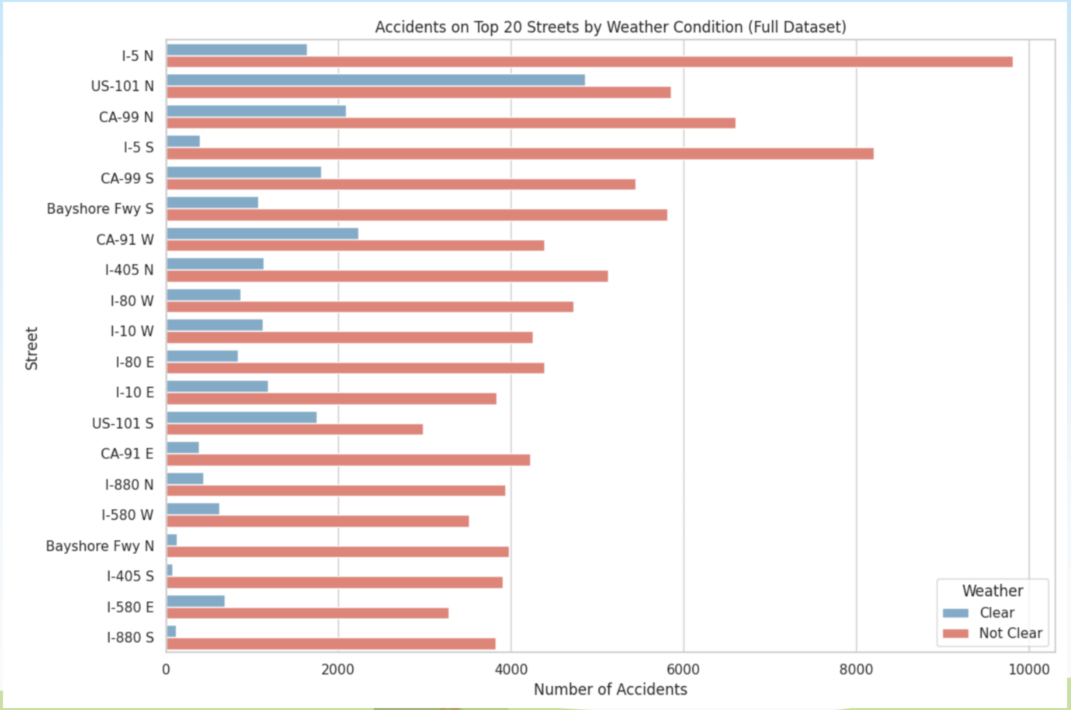
Descriptions have high cardinality, bucketed into 6 binary flags - 'Is_Windy', 'Is_Stormy', 'Is_Rainy', 'Is_Foggy', 'Is_Snowy', 'Is_Clear'



	Temperature(F)	Humidity(%)	Pressure(in)	Wind_Speed(mph)	Severity
Temperature(F)	1	-0.65	0.042	0.21	0.046
Humidity(%)	-0.65	1	0.12	-0.2	0.027
Pressure(in)	0.042	0.12	1	-0.018	0.1
Wind_Speed(mph)	0.21	-0.2	-0.018	1	0.045
Severity	0.046	0.027	0.1	0.045	1

Weather Analysis

Generally we see that unclear weather contributes heavily to the occurrence of accidents, making it a key factor that we wish to consider. The different factors of weather can be analysed to make predictions.



EDA Conclusions

Regions

Base core analysis on the cities and streets (could have issues), as they **show trends** and also are the most **actionable**

Day and Time

Do analysis based on the **peak hours** of the day, and whether it is a **weekday or weekend**

Weather Conditions

Consider all **non-clear weather** conditions, **less** so for specific **roads with a homogeneous split**

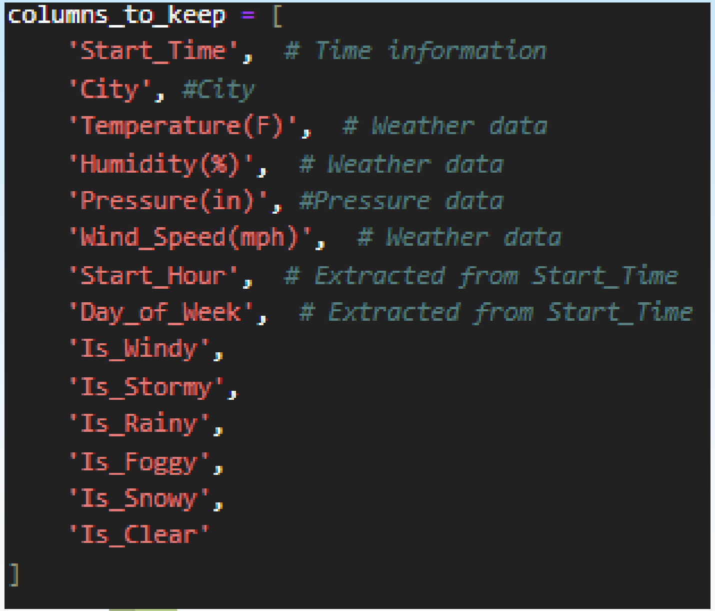
Machine Learning

There were 3 approaches for our ML algorithm:

Accident Classification (Classification)	Rate of Accidents Prediction (Regression)	Severity of Accident (Classification)
Not possible as we possess a **positive-only dataset**, leading us to be unable to distinguish between accidents and non-accidents ✗	Possible use of linear regression to determine the rate of accidents in a city over the next hour based on additional information such as time, weather and location ✓	Extremely low quantities of high severity data, resulting in imbalance in the data set ✗

Reducing Dataframe Size

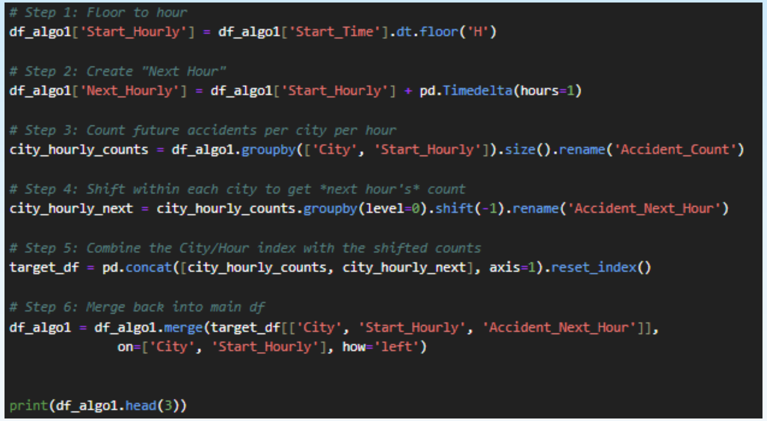
Based on the EDA conducted, we have identified a few key metrics that could affect the accident rates. From factors such as the **City, Day of the Week as well as Weather condition**. These variables will be extracted to trim the dataset for faster processing by the algorithms. Street is removed here due to its extremely high cardinality, instead focusing on City.



```
columns_to_keep = [
    'Start_Time',  # Time information
    'City', #City
    'Temperature(F)',  # Weather data
    'Humidity(%)',  # Weather data
    'Pressure(in)', #Pressure data
    'Wind_Speed(mph)',  # Weather data
    'Start_Hour',  # Extracted from Start_Time
    'Day_of_Week',  # Extracted from Start_Time
    'Is_Windy',
    'Is_Stormy',
    'Is_Rainy',
    'Is_Foggy',
    'Is_Snowy',
    'Is_Clear'
]
```

Further Feature Engineering

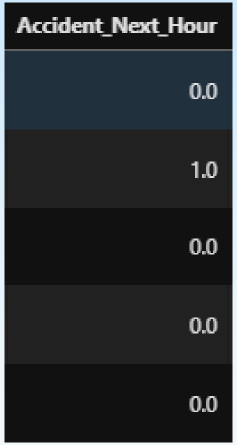
Feature Creation



```
# Step 1: Floor to hour
df_algo1['Start_Hourly'] = df_algo1['Start_Time'].dt.floor('H')

# Step 2: Create "Next Hour"
df_algo1['Next_Hourly'] = df_algo1['Start_Hourly'] + pd.Timedelta(hours=1)

# Step 3: Count future accidents per city per hour
city_hourly_counts = df_algo1.groupby(['City', 'Start_Hourly']).size().rename('Accident_Count')

# Step 4: Shift within each city to get *next hour's* count
city_hourly_next = city_hourly_counts.groupby(level=0).shift(-1).rename('Accident_Next_Hour')

# Step 5: Combine the City/Hour index with the shifted counts
target_df = pd.concat([city_hourly_counts, city_hourly_next], axis=1).reset_index()

# Step 6: Merge back into main df
df_algo1 = df_algo1.merge(target_df[['City', 'Start_Hourly', 'Accident_Next_Hour']],
              on=['City', 'Start_Hourly'], how='left')


print(df_algo1.head(3))
```

Created a new feature 'Accident_Next_Hour' based on the city and hour as an indicator of the accidents occuring in the next hour

Target Variable

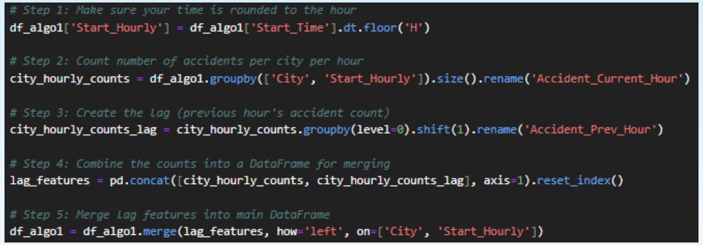


Accident_Next_Hour
0.0
1.0
0.0
0.0
0.0

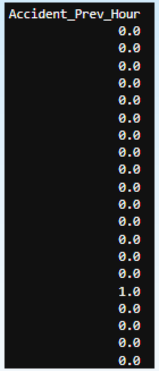
This new feature would act as the target variable for our models to predict

Further Feature Engineering

Feature Creation



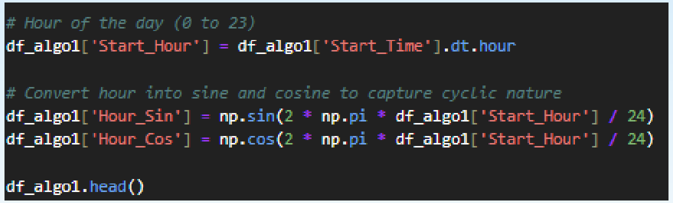
```python
# Step 1: Make sure your time is rounded to the hour
df_algo1['Start_Hourly'] = df_algo1['Start_Time'].dt.floor('H')

# Step 2: Count number of accidents per city per hour
city_hourly_counts = df_algo1.groupby(['City', 'Start_Hourly']).size().rename('Accident_Current_Hour')

# Step 3: Create the lag (previous hour's accident count)
city_hourly_counts_lag = city_hourly_counts.groupby(level=0).shift(1).rename('Accident_Prev_Hour')

# Step 4: Combine the counts into a DataFrame for merging
lag_features = pd.concat([city_hourly_counts, city_hourly_counts_lag], axis=1).reset_index()

# Step 5: Merge lag features into main DataFrame
df_algo1 = df_algo1.merge(lag_features, how='left', on=['City', 'Start_Hourly'])
```



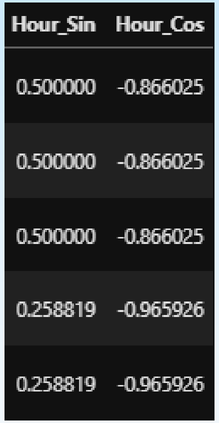
```
Accident_Prev_Hour
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               0.0
               1.0
               0.0
               0.0
               0.0
               0.0
```

Created a new feature ‘Accident_Prev_Hour’ based on the city and hour as an indicator of the accidents occuring in the previous hour, to feed time lag data into our model

Feature Creation



```python
# Hour of the day (0 to 23)
df_algo1['Start_Hour'] = df_algo1['Start_Time'].dt.hour

# Convert hour into sine and cosine to capture cyclic nature
df_algo1['Hour_Sin'] = np.sin(2 * np.pi * df_algo1['Start_Hour'] / 24)
df_algo1['Hour_Cos'] = np.cos(2 * np.pi * df_algo1['Start_Hour'] / 24)

df_algo1.head()
```

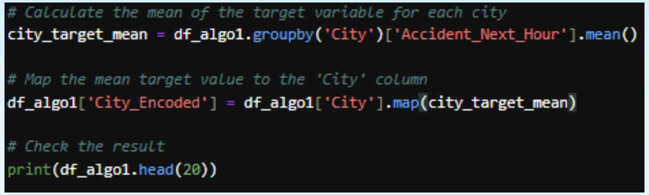


Hour_Sin	Hour_Cos
0.500000	-0.866025
0.500000	-0.866025
0.500000	-0.866025
0.258819	-0.965926
0.258819	-0.965926

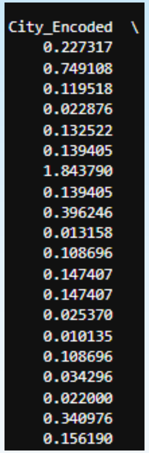
Created new features ‘Hour_Sin’ and ‘Hour_Cos’ to better reflect the cylic nature of the hours of the day

Data Transformation

Target Encoding

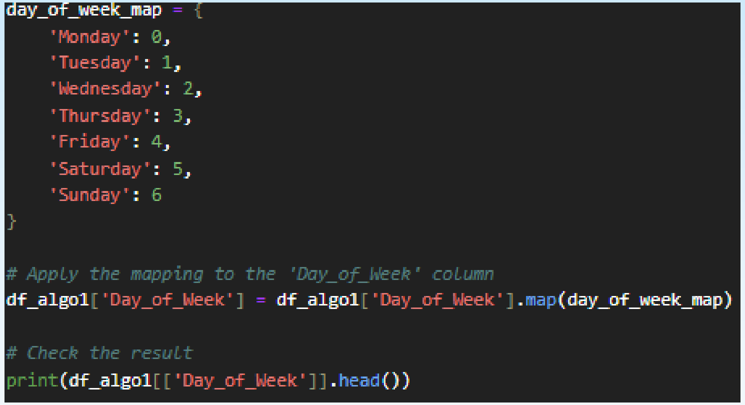


```python
# Calculate the mean of the target variable for each city
city_target_mean = df_algo1.groupby('City')['Accident_Next_Hour'].mean()

# Map the mean target value to the 'City' column
df_algo1['City_Encoded'] = df_algo1['City'].map(city_target_mean)

# Check the result
print(df_algo1.head(20))
```



```
City_Encoded  \
    0.227317
    0.749108
    0.119518
    0.022876
    0.132522
    0.139405
    1.843790
    0.139405
    0.396246
    0.013158
    0.108696
    0.147407
    0.147407
    0.025370
    0.010135
    0.108696
    0.034296
    0.022000
    0.340976
    0.156190
```

Ordinal Encoding

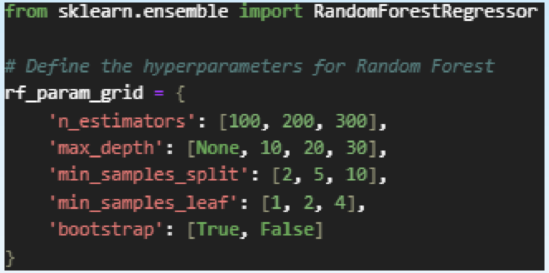


```python
day_of_week_map = {
    'Monday': 0,
    'Tuesday': 1,
    'Wednesday': 2,
    'Thursday': 3,
    'Friday': 4,
    'Saturday': 5,
    'Sunday': 6
}

# Apply the mapping to the 'Day_of_Week' column
df_algo1['Day_of_Week'] = df_algo1['Day_of_Week'].map(day_of_week_map)

# Check the result
print(df_algo1[['Day_of_Week']].head())
```

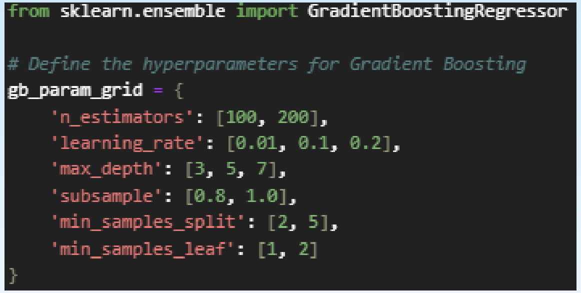
Due to the **high cardinality** of the City variable, One Hot Encoding was not possible, while Label Encoding could introduce unintended patterns, use **Target Encoding**

Since Day_Of_Week has an inherent order, it can be converted by undergoing a simple **Ordinal Encoding**

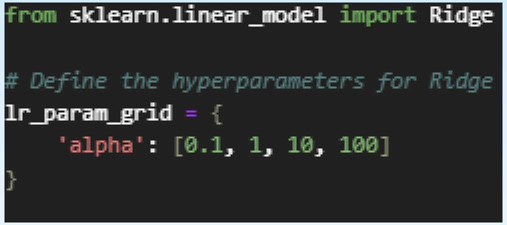
Hyperparameter Tuning



```python
from sklearn.ensemble import RandomForestRegressor

# Define the hyperparameters for Random Forest
rf_param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
```



```python
from sklearn.ensemble import GradientBoostingRegressor

# Define the hyperparameters for Gradient Boosting
gb_param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 1.0],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}
```



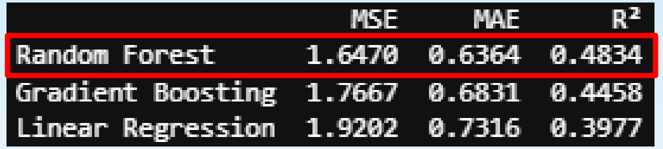
```python
from sklearn.linear_model import Ridge

# Define the hyperparameters for Ridge
lr_param_grid = {
    'alpha': [0.1, 1, 10, 100]
}
```

Random Forest

Gradient Boosting

Linear Regression

Model Results



```
                     MSE     MAE      R²
Random Forest      1.6470  0.6364  0.4834
Gradient Boosting  1.7667  0.6831  0.4458
Linear Regression  1.9202  0.7316  0.3977
```

From our results, it is clear that Random Forest Regressor performed the best with the lowest MSE and MAE and highest R^2

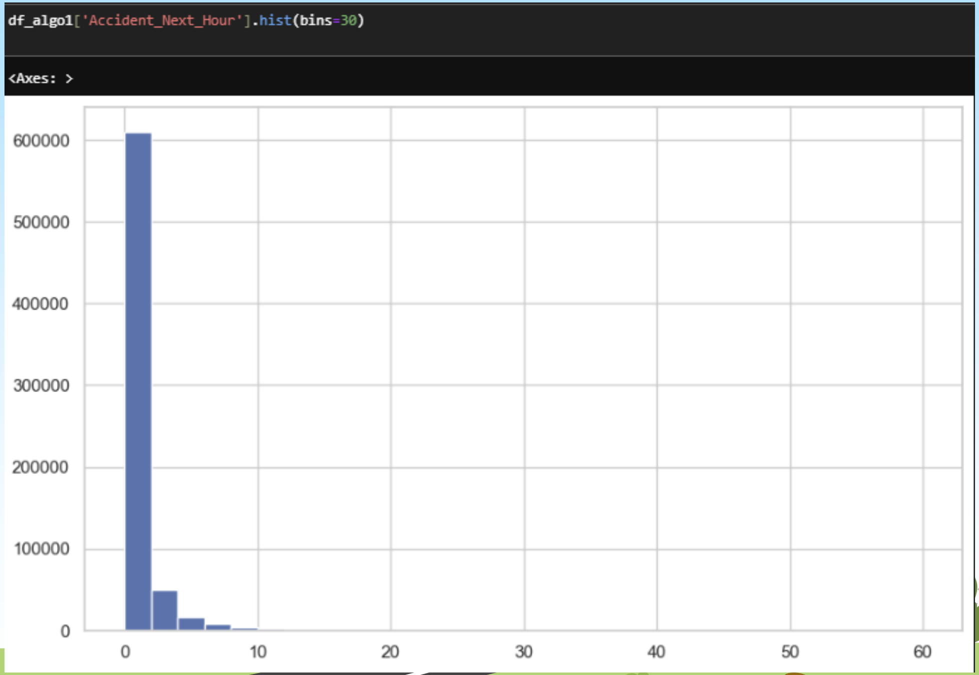
Model Results

But why is the R^2 value of the model still so low?

From our results, it is clear that Random Forest Regressor performed the best with the lowest MSE and MAE and highest R^2

Model Results

Looking at the bins of possible accidents in the next hour, it is clear that the majority lie in the range of 0-2. As a result, the model could be overfitting as it cannot account for the outliers in the regression. Could converting this into a classification problem be better?

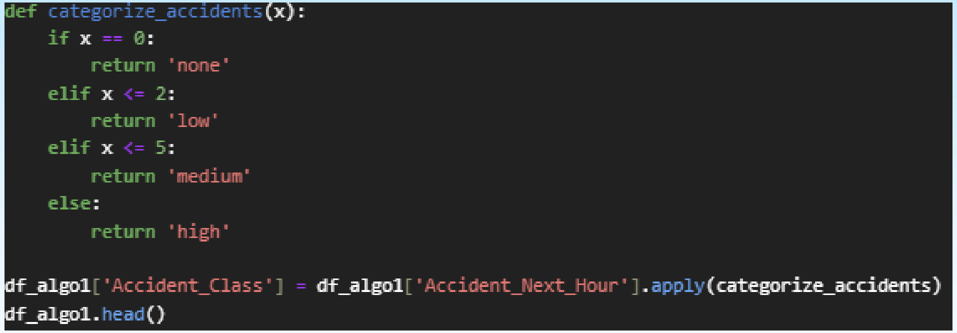


```
df_algo1['Accident_Next_Hour'].hist(bins=30)
```

```
<Axes: >
```

Model Results

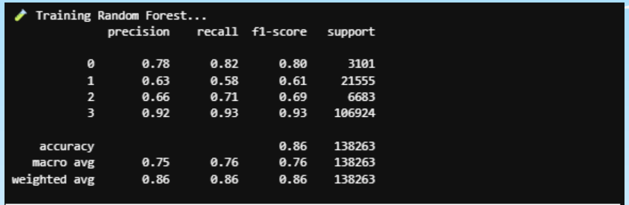
Using a different approach, we now classify the rate of accidents into 4 buckets, none, low, medium and high.



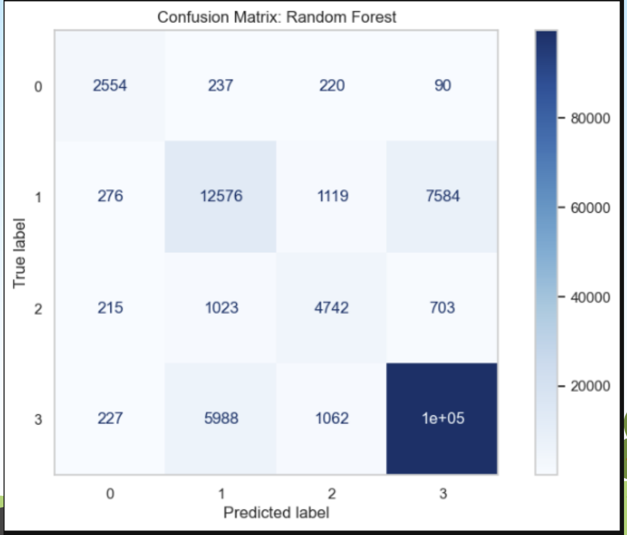
```
def categorize_accidents(x):
    if x == 0:
        return 'none'
    elif x <= 2:
        return 'low'
    elif x <= 5:
        return 'medium'
    else:
        return 'high'

df_algo1['Accident_Class'] = df_algo1['Accident_Next_Hour'].apply(categorize_accidents)
df_algo1.head()
```

Model Results

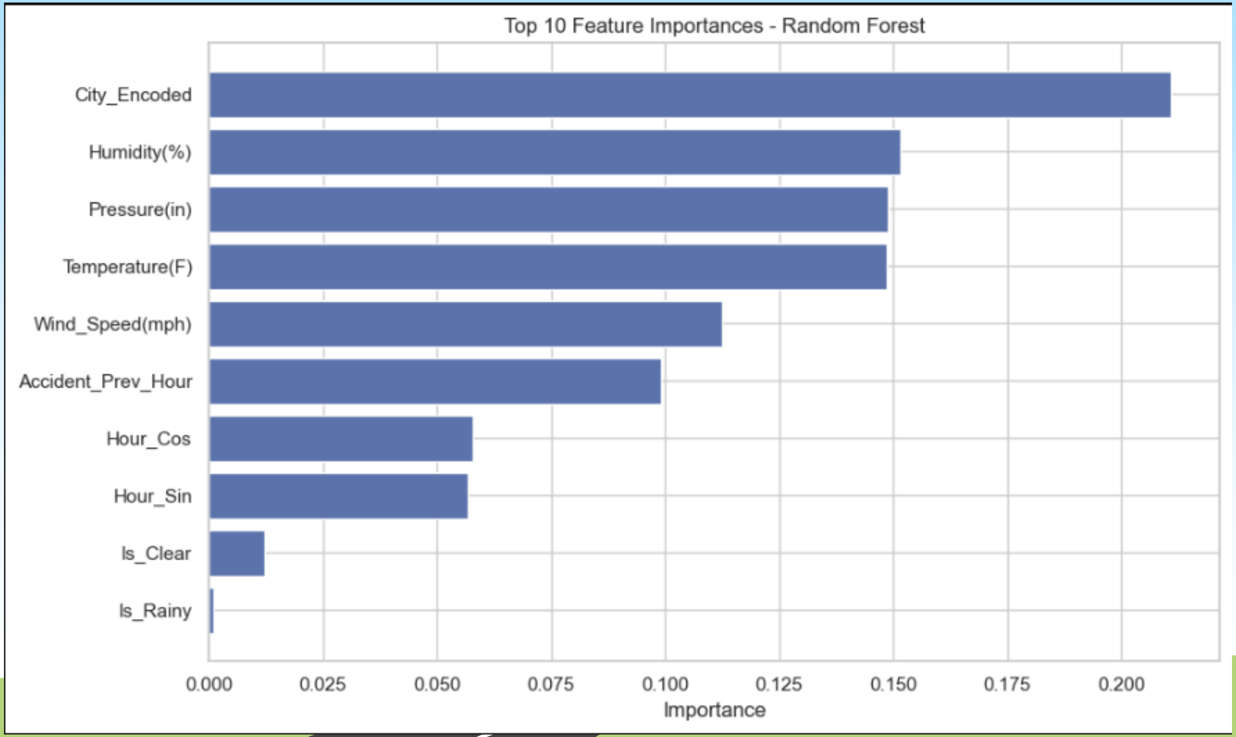
Out of Random Forest, Gradient Boosting, Logistic Regression, XGBoost and HistGradientBoosting, **Random Forest performed the best**. It could quite accurately predict no accidents (3) with F1 score of 0.93 and high rates of accidents (0) with F1 score of 0.80, but struggled with predicting in between, at low (2) and especially as medium (1) rates of accidents.



```
 Training Random Forest...
              precision    recall  f1-score   support

           0       0.78      0.82      0.80      3101
           1       0.63      0.58      0.61     21555
           2       0.66      0.71      0.69      6683
           3       0.92      0.93      0.93    106924

    accuracy                           0.86    138263
   macro avg       0.75      0.76      0.76    138263
weighted avg       0.86      0.86      0.86    138263
```



Feature Importance



Looking at the feature importance for the Random Forest, it is clear that City is the greatest predictor of the rate of accident.

Conclusion

Problem Addressing

Unable to predict numeric probability of accidents due to limitations (lack of negative data), but identified key determinants to look at

Interesting Insights

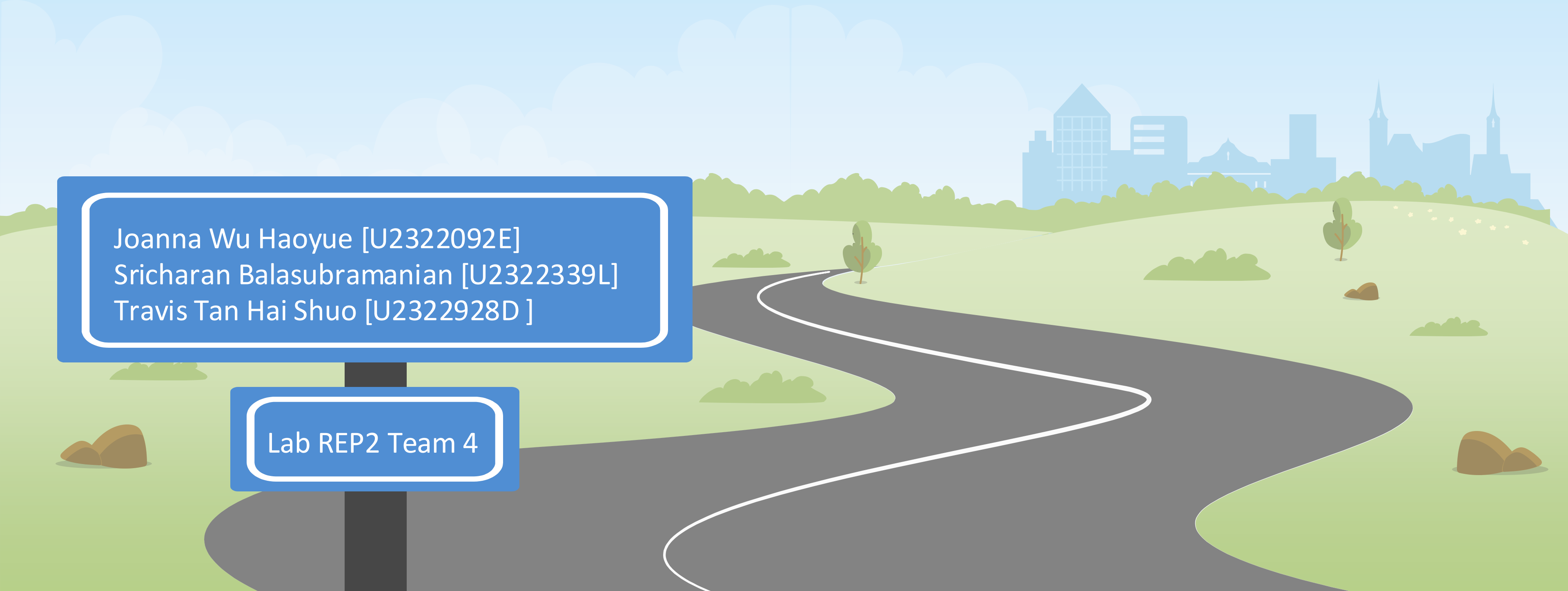
Oddly, humidity seems to have the largest impact on the rate of accidents other than the city we are looking at, while conditions like pressure also seem to have large impacts

Use Case and Future Scope

Use in our own road trips in California

Import traffic details for live inputs and negative data as we drive

Thank you!



Joanna Wu Haoyue [U2322092E]
Sricharan Balasubramanian [U2322339L]
Travis Tan Hai Shuo [U2322928D]

Lab REP2 Team 4