Bicycles Accident Analysis

Mohammad Nurrokim

Introduction: Business Problem

This report will try to analyze the best location and time for cycling activities. Specifically, this project will target stakeholders interested in **cycling activities** such as individual cyclists, cycling communities, and companies/sponsors/event organizers of cycling activities.

During a pandemic, many people choose cycling as an alternative to sports. Cycling is considered the safest way to exercise because of minimal contact with other people. There are many accidents involving cyclists. Cyclists also need protection and a sense of security while on the road. Cycling is not only interpreted as transportation activity, but also sports and recreational activities.

When an accident occurs, car drivers are still protected by car frames and car safety technology in comparison. So, the chances of surviving or being injured are still relatively low compared to cyclists. Cyclists are only protected by wearing helmets on their heads. When an accident occurs, their bodies, feet, and hands have the potential to be injured.

This project will assist the Seattle Department of Transportation (SDOT) to provide different traffic signs in accident-prone areas for cyclists. This project will also help the cyclist community like **Cascade Bicylcle Club, COGS (Cyclists Of Greater Seattle), Brake the Cycle, etc.** to find out the right track and time to hold a cycling event.

Many events are held by many cyclist communities, like Cascade Bicycle Club, for example.. This club hosts **several major riding events** every year including Chilly Hilly, Seattle Bike-n-Brews, Ride for Major Taylor, Flying Wheels Summer Century, Woodinville Wine Ride, Seattle Night Ride, the Red-Bell 100, Seattle to Portland (STP), Ride from Seattle to Vancouver and Party (RSVP), Ride Around Washington (RAW), High Pass Challenge (HPC), and Kitsap Color Classic (KCC) (Wikipedia). This project can help **companies, sponsors, and event organizers** to create **safe cycling events** for all participants.

```
data = pd.read_csv('Data-Collisions.csv')
```

Data

Based on definition of our problem, features or columns that will influence our analysis are:

1. **Location**: Latitude (X), Longitude (Y), Address Type (ADDRTYPE)
2. **Severity**: A code that corresponds to the severity of the collision (SEVERITYCODE), a detailed description of the severity of the collision (SEVERITYDESC)
3. **Person Count**: Total number of people involved (PERSONCOUNT), number of bicycles involved in the collision (PEDCYLCOUNT)
4. **Date**: The date and time of the incident (INCDTTM)
5. **Condition**: Description of the weather conditions (WEATHER), condition of the road (ROADCOND), light conditions during the collision (LIGHTCOND)

Conditional Selection — Only show data that bicycles involved in the collision

In the explanation of the problem above, we will help solve the problem for cyclists. We limit data on car accidents involving cyclists. So the PEDCYLCOUNT column must be greater than zero.

```
# Conditional selection, column 'PEDCYLCOUNT' greater than 0
data = data[data['PEDCYLCOUNT']>0].reset_index()
data.head(3)
```

DataFrame Shape

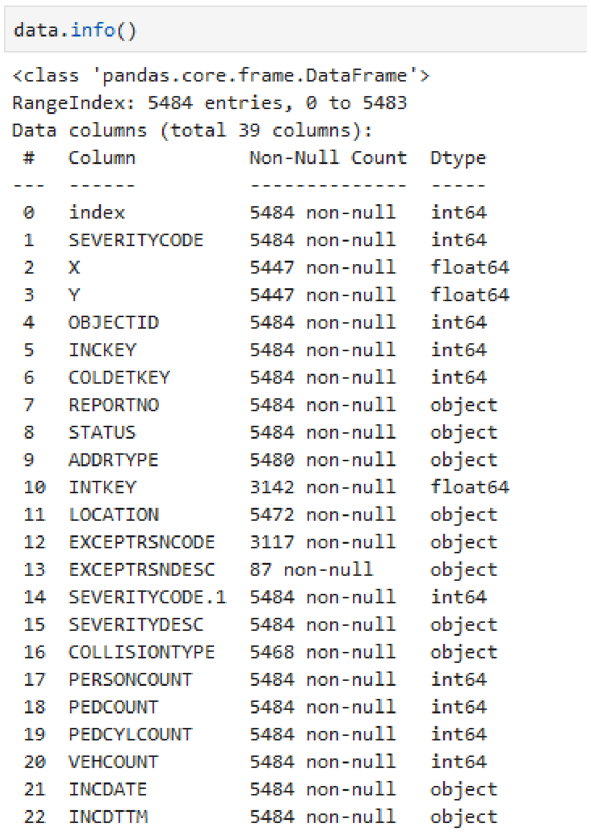
The dataset used is (5484 rows, 38 columns). Not all columns will be used, will be selected according to the data description above.

```
data.shape
```

```
(5484, 39)
```

DataFrame Data Type and Missing Values

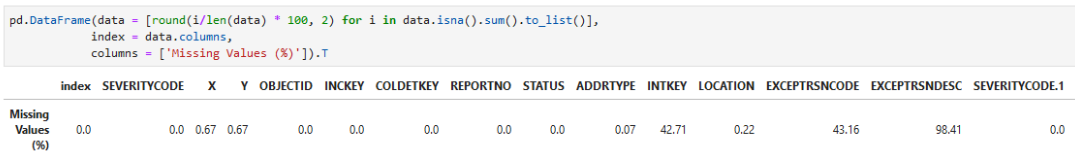
In the description below, we will know the data types and missing values and their presentations.



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5484 entries, 0 to 5483
Data columns (total 39 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   index           5484 non-null   int64  
 1   SEVERITYCODE    5484 non-null   int64  
 2   X               5447 non-null   float64
 3   Y               5447 non-null   float64
 4   OBJECTID        5484 non-null   int64  
 5   INCKEY          5484 non-null   int64  
 6   COLDETKEY       5484 non-null   int64  
 7   REPORTNO        5484 non-null   object 
 8   STATUS          5484 non-null   object 
 9   ADDRTYPE        5480 non-null   object 
 10  INTKEY          3142 non-null   float64
 11  LOCATION        5472 non-null   object 
 12  EXCEPTRSNCODE   3117 non-null   object 
 13  EXCEPTRSNDESC   87 non-null     object 
 14  SEVERITYCODE.1  5484 non-null   int64  
 15  SEVERITYDESC    5484 non-null   object 
 16  COLLISIONTYPE   5468 non-null   object 
 17  PERSONCOUNT     5484 non-null   int64  
 18  PEDCOUNT        5484 non-null   int64  
 19  PEDCYLCOUNT     5484 non-null   int64  
 20  VEHCOUNT        5484 non-null   int64  
 21  INCDATE         5484 non-null   object 
 22  INCDTTM         5484 non-null   object 
```

Missing Values Percentage



```
pd.DataFrame(data = [round(i/len(data) * 100, 2) for i in data.isna().sum().to_list()],
            index = data.columns,
            columns = ['Missing Values (%)']).T
```

	index	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	LOCATION	EXCEPTRSNCODE	EXCEPTRSNDESC	SEVERITYCODE.1
Missing Values (%)	0.0	0.0	0.67	0.67	0.0	0.0	0.0	0.0	0.0	0.07	42.71	0.22	43.16	98.41	0.0

Methodology

1. Features Selection

Not all features are used for analysis in this project. Thus, only some data is displayed and analyzed.

2. Handling Missing Values

Missing values will interfere with the prediction and analysis results. So, we need to handle the missing values by deleting them or filling them in. If there are not too many missing values, we can choose the option to delete them.

3. Handling Duplicates Values

Duplicate values will also interfere with the analysis and prediction results. First, we need to detect the number of duplicate values in the dataset. Next, these duplicate values need to be removed to make the dataset cleaner.

4. Convert 'INCDTTM' Column to Datetime Type

'ICDDTM' Column needs to be changed in the DateTime type. Because by converting it to a DateTime type, we can extract hour, day, month, and year data. These data can help us to analyze data more deeply.

5. Exploratory Data Analysis (EDA)

After cleaning the data, we can run the exploratory data analysis. The analysis framework follows the problem we have defined, namely finding the best time and location for cycling activities.

First, the data will be explored and analyzed based on data related to time, such as the hour, day, month, year, and weather. The data is visualized to get an overview of the best time to hold a cycling event. Second, looking for an overview of the conditions for the best place to hold a cycling event. The data visualized include light conditions, road conditions, and address types.

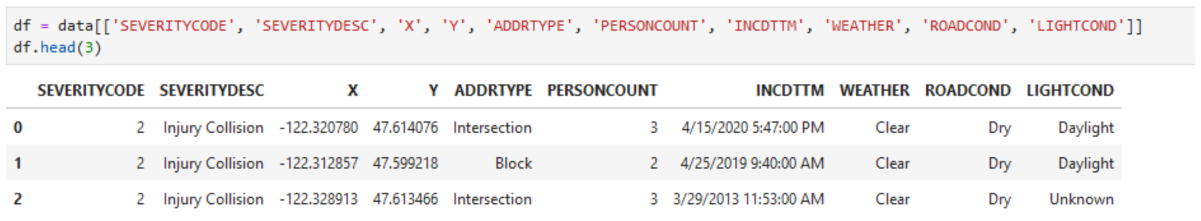
6. Model Building

The machine learning model used in this project is logistic regression. Why use logistic regression? First, the data is binary. Second, we need probabilistic results to find out the time and place conditions that are most likely to cause injury collision. Before model building, the data will be encoded using the one-hot encoder and split into training and testing data.

Analysis

Features Selection

We use features that give support to solve the problems that have been planned.

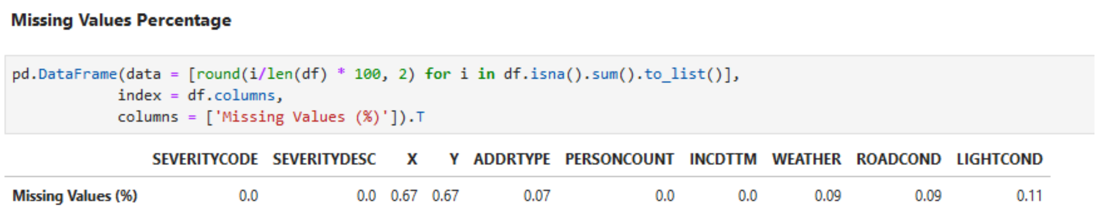


```
df = data[['SEVERITYCODE', 'SEVERITYDESC', 'X', 'Y', 'ADDRTYPE', 'PERSONCOUNT', 'INCDTTM', 'WEATHER', 'ROADCOND', 'LIGHTCOND']]
df.head(3)
```

	SEVERITYCODE	SEVERITYDESC	X	Y	ADDRTYPE	PERSONCOUNT	INCDTTM	WEATHER	ROADCOND	LIGHTCOND
0	2	Injury Collision	-122.320780	47.614076	Intersection	3	4/15/2020 5:47:00 PM	Clear	Dry	Daylight
1	2	Injury Collision	-122.312857	47.599218	Block	2	4/25/2019 9:40:00 AM	Clear	Dry	Daylight
2	2	Injury Collision	-122.328913	47.613466	Intersection	3	3/29/2013 11:53:00 AM	Clear	Dry	Unknown

Handling Missing Values

In the table below, we can see that the missing values in all features are below 1 per cent.

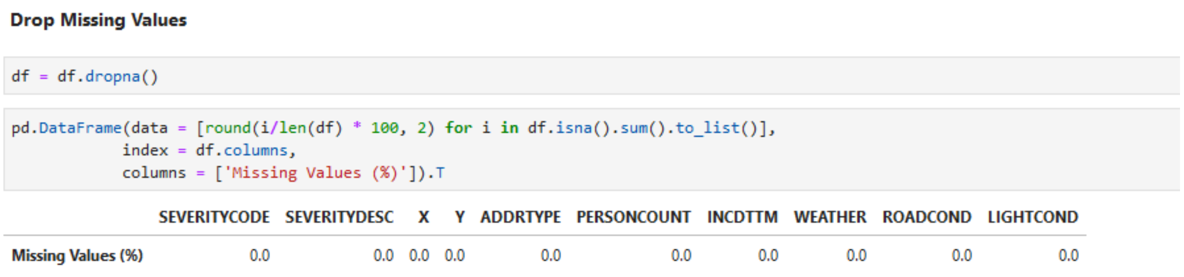


Missing Values Percentage

```
pd.DataFrame(data = [round(i/len(df) * 100, 2) for i in df.isna().sum().to_list()],
             index = df.columns,
             columns = ['Missing Values (%)']).T
```

	SEVERITYCODE	SEVERITYDESC	X	Y	ADDRTYPE	PERSONCOUNT	INCDTTM	WEATHER	ROADCOND	LIGHTCOND
Missing Values (%)	0.0	0.0	0.67	0.67	0.07	0.0	0.0	0.09	0.09	0.11

Because the missing values are minimal (less than 1 per cent), we can decide to delete the missing values in all features.



Drop Missing Values

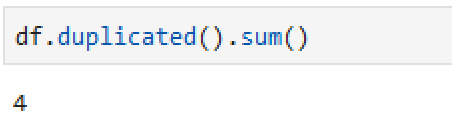
```
df = df.dropna()
```

```
pd.DataFrame(data = [round(i/len(df) * 100, 2) for i in df.isna().sum().to_list()],
             index = df.columns,
             columns = ['Missing Values (%)']).T
```

	SEVERITYCODE	SEVERITYDESC	X	Y	ADDRTYPE	PERSONCOUNT	INCDTTM	WEATHER	ROADCOND	LIGHTCOND
Missing Values (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Handling Duplicate Values

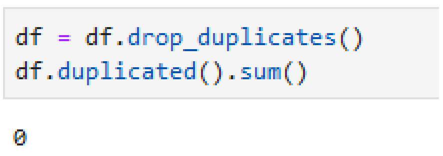
We also need to check the data if there is duplicate data. Many duplicate values will affect the analysis and prediction results. Therefore we need to detect the amount of duplicate data. Then we delete the duplicate value.



```
df.duplicated().sum()
```

```
4
```

In the output above, there are four duplicate data. After that, we drop these duplicate values from the dataset. So, we get a cleaner dataset for analysis.

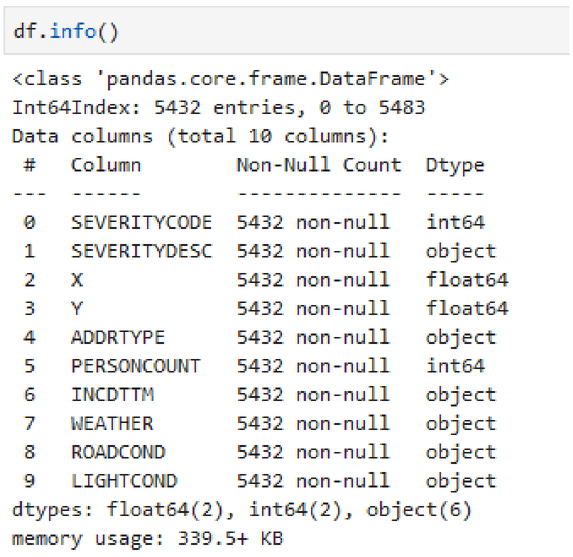


```
df = df.drop_duplicates()
df.duplicated().sum()
```

```
0
```

Convert 'INCDTTM' Column to DateTime Type

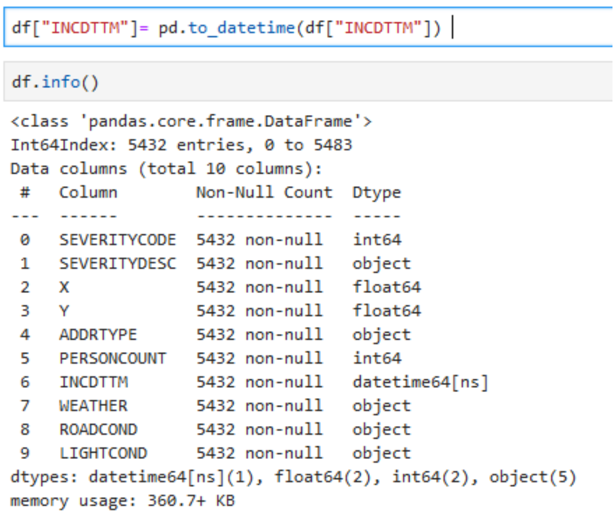
Below, we can see that the ‘INCDTTM’ Column type is still an ‘object’.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5432 entries, 0 to 5483
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   SEVERITYCODE  5432 non-null   int64  
 1   SEVERITYDESC  5432 non-null   object 
 2   X             5432 non-null   float64
 3   Y             5432 non-null   float64
 4   ADDRTYPE      5432 non-null   object 
 5   PERSONCOUNT   5432 non-null   int64  
 6   INCDTTM       5432 non-null   object 
 7   WEATHER       5432 non-null   object 
 8   ROADCOND      5432 non-null   object 
 9   LIGHTCOND     5432 non-null   object 
dtypes: float64(2), int64(2), object(6)
memory usage: 339.5+ KB
```

After changing to DateTime type, column ‘INCDTTM’ changes to ‘datetime64.’

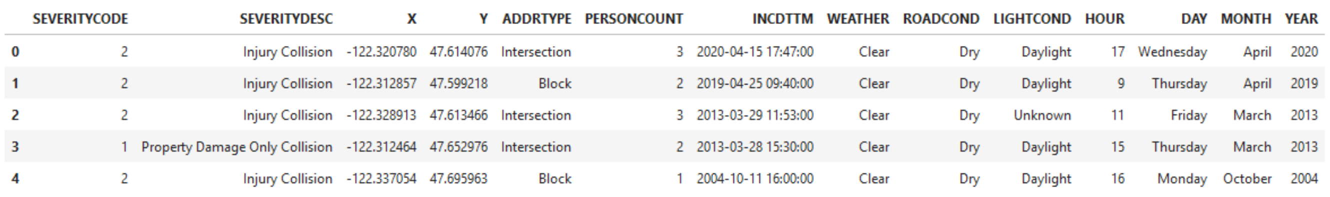


```
df["INCDTTM"]= pd.to_datetime(df["INCDTTM"])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5432 entries, 0 to 5483
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   SEVERITYCODE  5432 non-null   int64         
 1   SEVERITYDESC  5432 non-null   object        
 2   X             5432 non-null   float64       
 3   Y             5432 non-null   float64       
 4   ADDRTYPE      5432 non-null   object        
 5   PERSONCOUNT   5432 non-null   int64         
 6   INCDTTM       5432 non-null   datetime64[ns]
 7   WEATHER       5432 non-null   object        
 8   ROADCOND      5432 non-null   object        
 9   LIGHTCOND     5432 non-null   object        
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 360.7+ KB
```

After converting to ‘datetime64’, we can extract the data into new columns such as hour, day, month, and year.



	SEVERITYCODE	SEVERITYDESC	X	Y	ADDRTYPE	PERSONCOUNT	INCDTTM	WEATHER	ROADCOND	LIGHTCOND	HOUR	DAY	MONTH	YEAR
0	2	Injury Collision	-122.320780	47.614076	Intersection	3	2020-04-15 17:47:00	Clear	Dry	Daylight	17	Wednesday	April	2020
1	2	Injury Collision	-122.312857	47.599218	Block	2	2019-04-25 09:40:00	Clear	Dry	Daylight	9	Thursday	April	2019
2	2	Injury Collision	-122.328913	47.613466	Intersection	3	2013-03-29 11:53:00	Clear	Dry	Unknown	11	Friday	March	2013
3	1	Property Damage Only Collision	-122.312464	47.652976	Intersection	2	2013-03-28 15:30:00	Clear	Dry	Daylight	15	Thursday	March	2013
4	2	Injury Collision	-122.337054	47.695963	Block	1	2004-10-11 16:00:00	Clear	Dry	Daylight	16	Monday	October	2004

Explanatory Data Analysis

This Exploratory Data Analysis is divided into two parts. First, Time Analysis, which explores the dataset using time-related features.

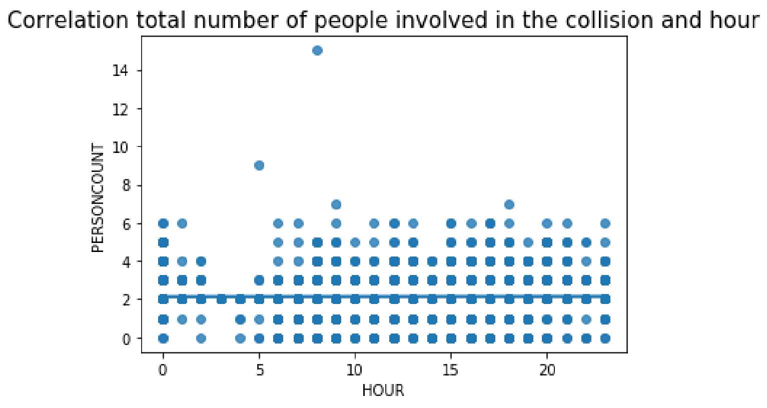
Second, Location Analysis, which explores the dataset using features related to roads, lighting, and location.

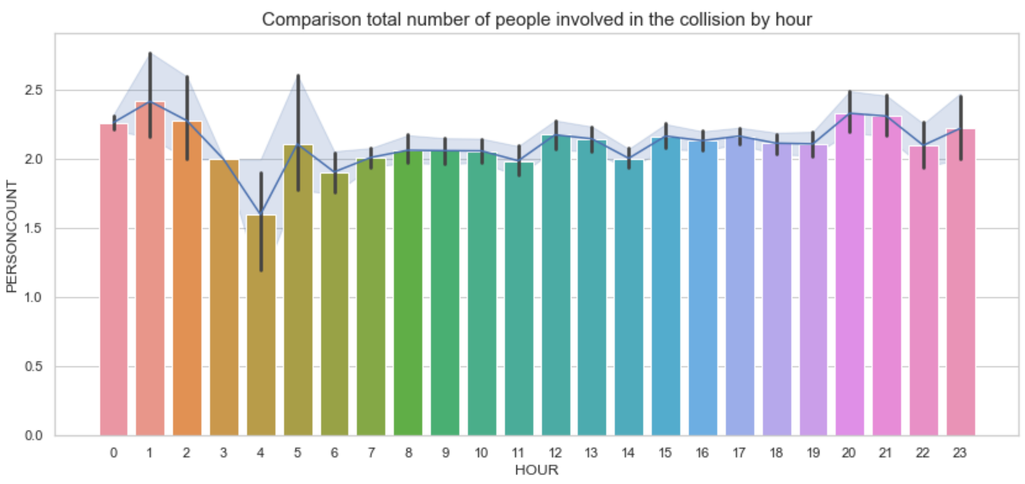
Time Analysis

In the visualization below, the hour variable is not related to the variable number of people involved in the collision.

Hour Variable

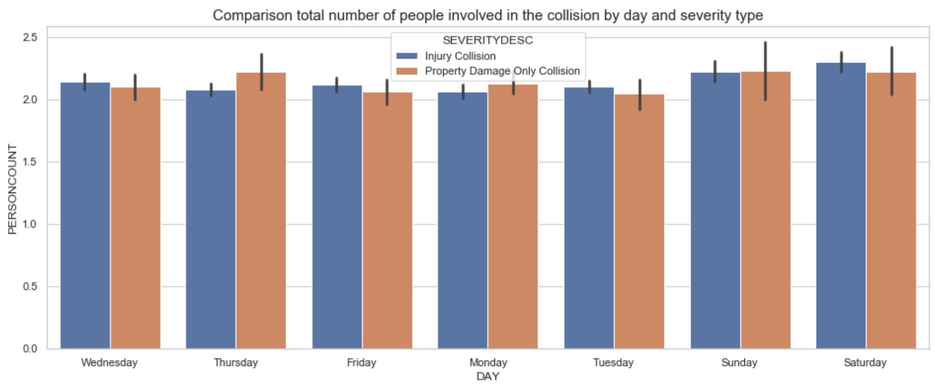
When tested for the correlation between hours and number of people involved, the score is only 0.004. This figure is shallow to qualify for a correlation (0.5).



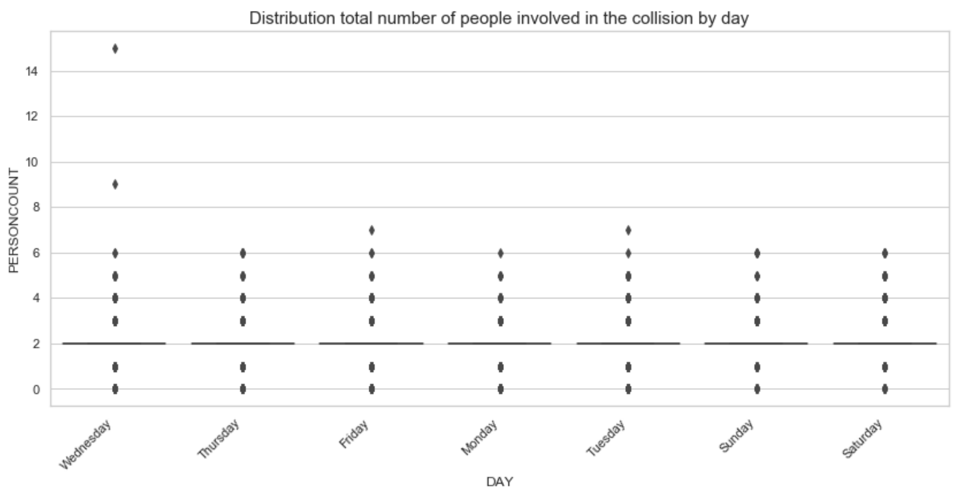


Day Variable

Below, visualize the total number of people involved in a collision by **day and severity type**.

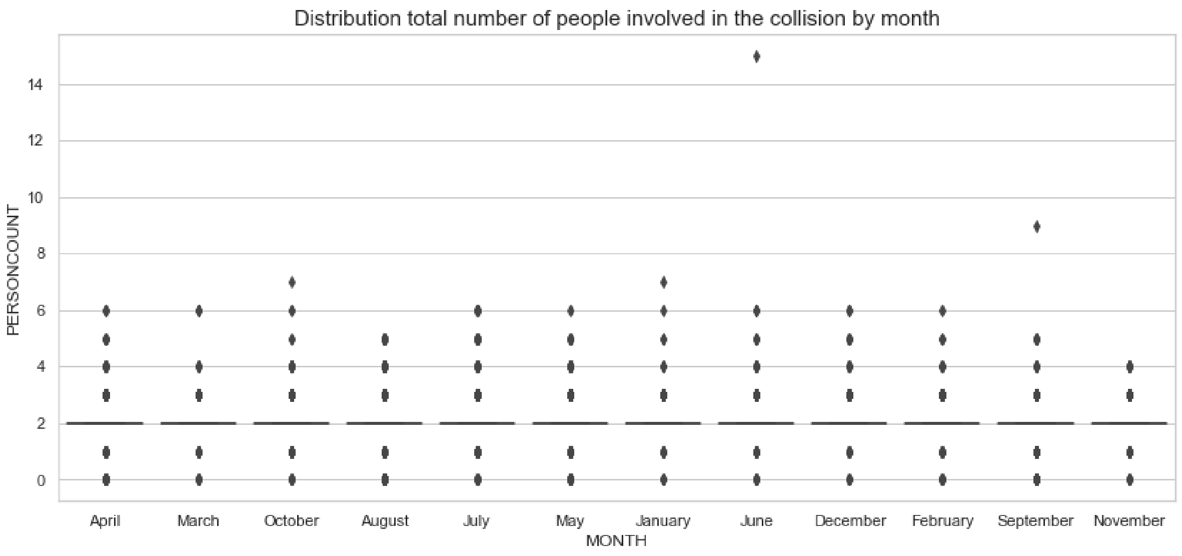


Below, a boxplot visualizes the distribution of the number of people involved in the collision by **day**. In this boxplot, we can also see the outliers in each data. The outlier data on **Wednesday** has the most considerable value in the data.



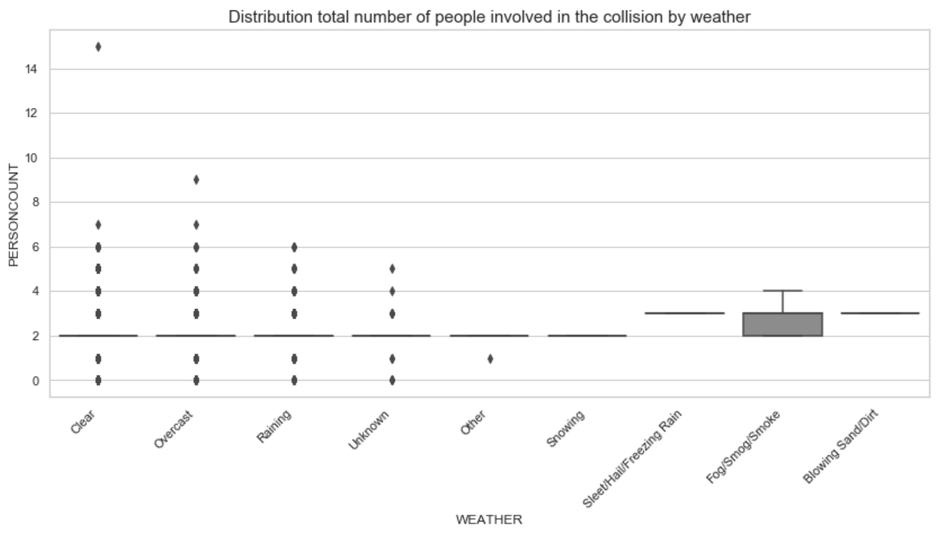
`Month` Variable

Below, a boxplot visualizes the distribution of the number of people involved in the collision by **month**. In this boxplot, we can also see the outliers in each data. The outlier data on **June** has the most considerable value in the data.



Weather Variable

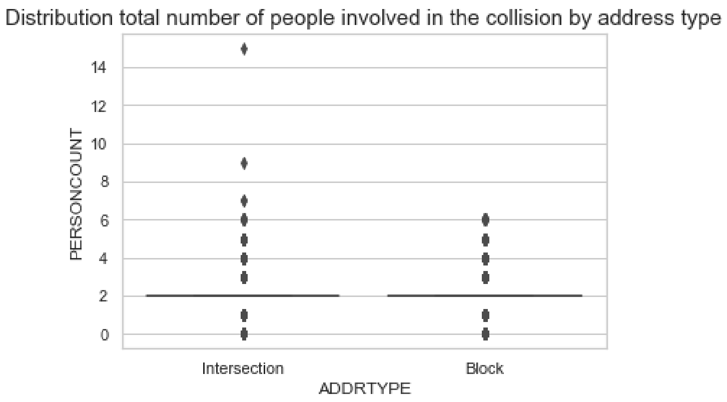
Below, a boxplot visualizes the distribution of the number of people involved in the collision by **weather**. In this boxplot, we can also see the outliers in each data.The outlier data on **Clear** weather has the most considerable value in the data.



Location & Condition Analysis

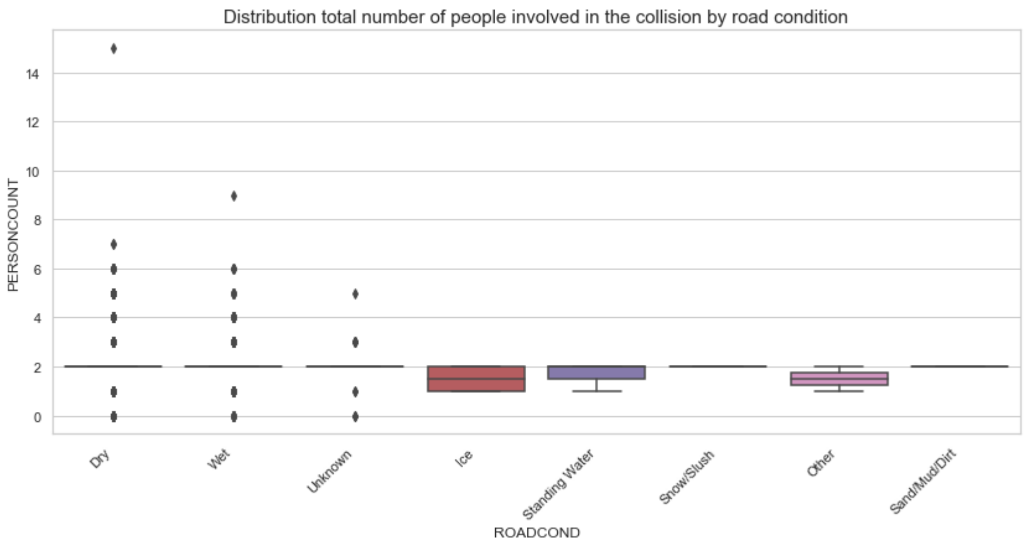
Address Type Variable

Below, a boxplot visualizes the distribution of the number of people involved in the collision by **address type**. In this boxplot, we can also see the outliers in each data. The outlier data on **Intersection** has the most considerable value in the data.



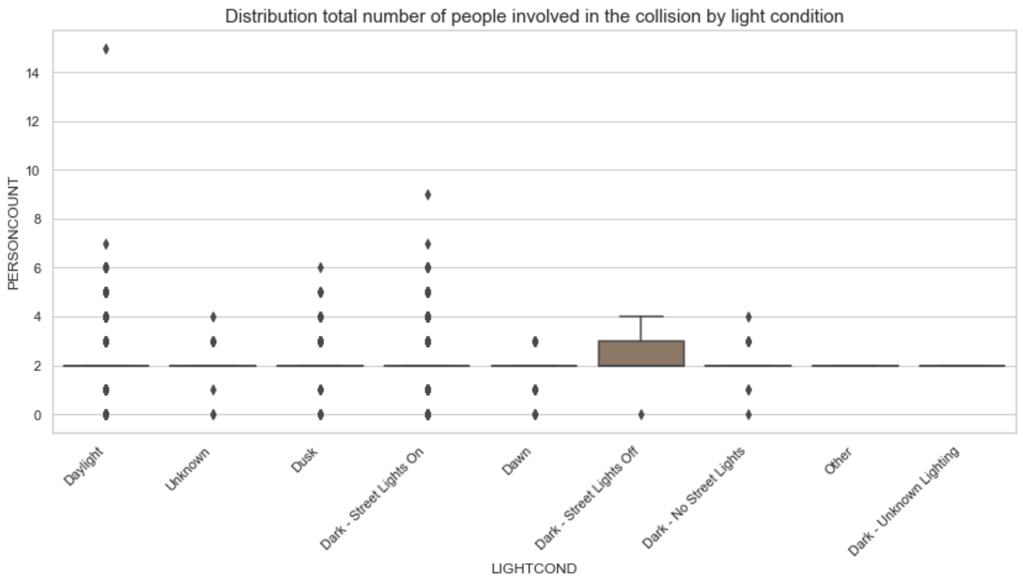
Road Condition Variable

Below, a boxplot visualizes the distribution of the number of people involved in the collision by **road condition**. In this boxplot, we can also see the outliers in each data. The outlier data on **Dry** condition has the most considerable value in the data.



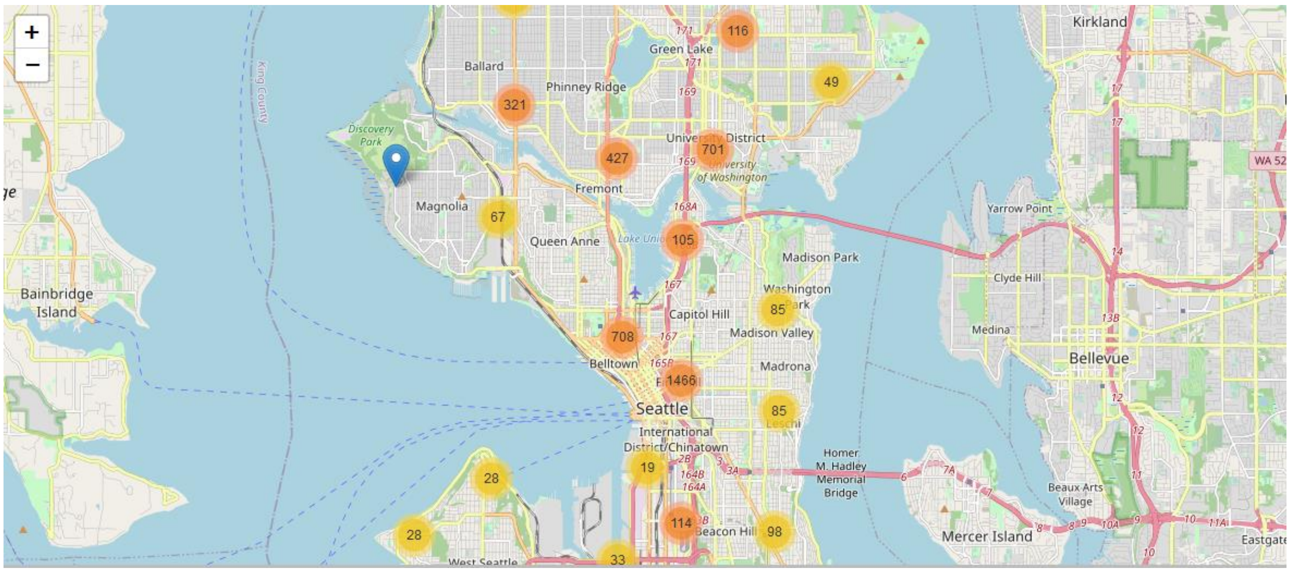
Light Condition Variable

Below, a boxplot visualizes the distribution of the number of people involved in the collision by **light condition**. In this boxplot, we can also see the outliers in each data.The outlier data on **Daylight** condition has the most considerable value in the data.



Collision Map that Involves Cyclists in Seattle

The map below was created using Folium. Using latitude and longitude features, we can recognize the number of collisions involving cyclists in Seattle. Also, if the map is expanded, details of the collision position will be observed.



Model Building Logistic Regression

What is Logistic Regression?

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

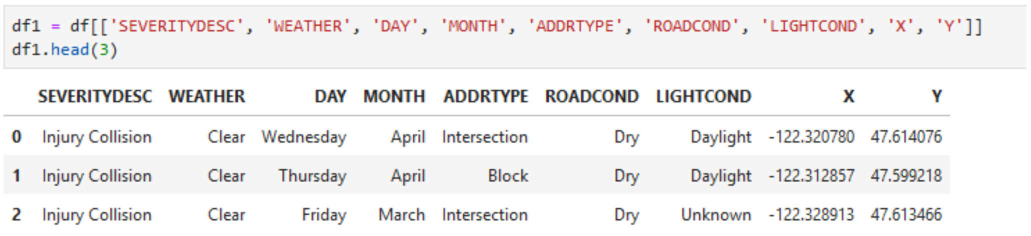
In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from logistic unit, hence the alternative names.

Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the odds ratio.

In a binary logistic regression model, the dependent variable has two levels (categorical). Outputs with more than two values are modeled by multinomial logistic regression and, if the multiple categories are ordered, by ordinal logistic regression (for example the proportional odds ordinal logistic model). The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other; this is a common way to make a binary classifier. The coefficients are generally not computed by a closed-form expression, unlike linear least squares. The logistic regression as a general statistical model was originally developed and popularized primarily by Joseph Berkson, beginning in Berkson (1944), where he coined "logit". **Source**: Wikipedia

Features Selection

Not all features are used for model building. The features below were selected for the Logistic Regression model training.

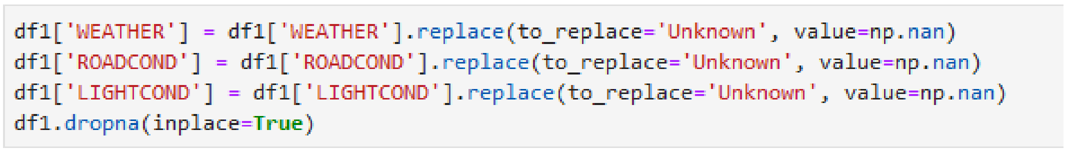


```python
df1 = df[['SEVERITYDESC', 'WEATHER', 'DAY', 'MONTH', 'ADDRTYPE', 'ROADCOND', 'LIGHTCOND', 'X', 'Y']]
df1.head(3)
```

	SEVERITYDESC	WEATHER	DAY	MONTH	ADDRTYPE	ROADCOND	LIGHTCOND	X	Y
0	Injury Collision	Clear	Wednesday	April	Intersection	Dry	Daylight	-122.320780	47.614076
1	Injury Collision	Clear	Thursday	April	Block	Dry	Daylight	-122.312857	47.599218
2	Injury Collision	Clear	Friday	March	Intersection	Dry	Unknown	-122.328913	47.613466

Cleaning 'Unknown' Values

The 'unknown' values in the WEATHER, LIGHTCOND, and ROADCOND fields need to be removed. This value needs to be deleted so that when using one hot encoder, this value is not used as a separate column. This is because the value of 'unknown' is difficult to understand.



```python
df1['WEATHER'] = df1['WEATHER'].replace(to_replace='Unknown', value=np.nan)
df1['ROADCOND'] = df1['ROADCOND'].replace(to_replace='Unknown', value=np.nan)
df1['LIGHTCOND'] = df1['LIGHTCOND'].replace(to_replace='Unknown', value=np.nan)
df1.dropna(inplace=True)
```

Splitting Dataset

Training & Test Sets

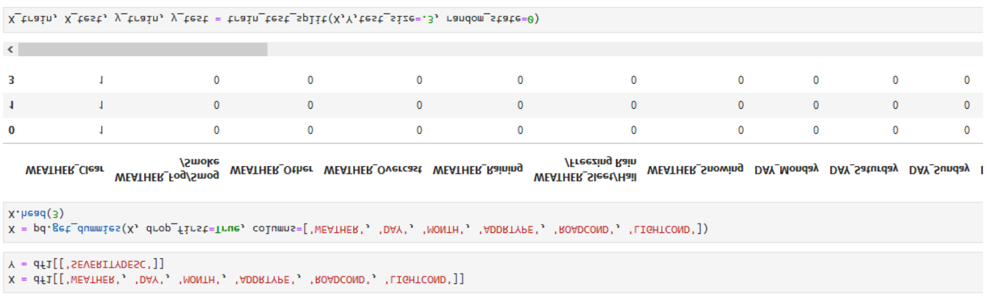
In machine learning, a common task is the study and construction of algorithms that can learn from and make predictions on data. Such algorithms function by making data-driven predictions or decisions, through building a mathematical model from input data. The data used to build the final model usually comes from multiple datasets. In particular, three datasets are commonly used in different stages of the creation of the model.

The model is initially fit on a **training dataset**, which is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model. The model (e.g. a neural net or a naive Bayes classifier) is trained on the training dataset using a supervised learning method, for example using optimization methods such as gradient descent or stochastic gradient descent. In practice, the training dataset often consists of pairs of an input vector (or scalar) and the corresponding output vector (or scalar), where the answer key is commonly denoted as the target (or label).

The current model is run with the training dataset and produces a result, which is then compared with the target, for each input vector in the training dataset. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

Finally, the **test dataset** is a dataset used to provide an unbiased evaluation of a final model fit on the training dataset. If the data in the test dataset has never been used in training (for example in cross-validation), the test dataset is also called a holdout dataset. Source: Wikipedia

First, the data is split into X (features / independent variable) and Y (target / dependent variable). Second, the data is encoded using the one-hot encoder. Finally, the dataset is split using a train-test-split function, with a composition of 30 percent test set and 70 percent training set.



```
X = dfl[['WEATHER', 'DAY', 'MONTH', 'ADDRTYPE', 'ROADCOND', 'LIGHTCOND']]
y = dfl[['SEVERITYDESC']]
```

```
X = pd.get_dummies(X, drop_first=True, columns=['WEATHER', 'DAY', 'MONTH', 'ADDRTYPE', 'ROADCOND', 'LIGHTCOND'])
X.head(3)
```

	WEATHER_Clear	WEATHER_Fog/Smog/Smoke	WEATHER_Other	WEATHER_Overcast	WEATHER_Raining	WEATHER_Sleet/Hail/Freezing Rain	WEATHER_Snowing	DAY_Monday	DAY_Saturday	DAY_Sunday
0	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.3, random_state=0)
```

Model Training

The Logistic Regression model is trained using the parameter C = 0.01, solver = ‘liblinear’.

```
LR = LogisticRegression(C=0.01,
solver='liblinear').fit(X_train,y_train)yhat = LR.predict(X_test)yhat_prob
= LR.predict_proba(X_test)
```

Model Evaluation

Jaccard Index

The Jaccard index, also known as Intersection over Union and the Jaccard similarity coefficient (originally given the French name coefficient de communauté by Paul Jaccard), is a statistic used for gauging the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. The Jaccard distance, which measures dissimilarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union. Source: Wikipedia

Jaccard similarity score in this project is 0.88. It’s a relatively high result.

```
jaccard_similarity_score(y_test, yhat)0.8853704876504117
```

Log Loss

In machine learning and mathematical optimization, loss functions for classification are computationally feasible loss functions representing the price paid for inaccuracy of predictions in classification problems (problems of identifying which category a particular observation belongs to). Log loss(Logarithmic loss) measures the performance of a classifier where the predicted output is a probability value between 0 and 1. Source: Wikipedia

Log loss score in this project is 0.36. It’s a relatively low result.

```
log_loss(y_test, yhat_prob)0.36069182774186537
```

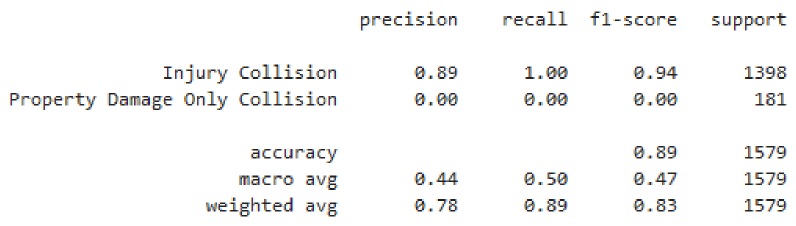
Classification Report

The classification report builds a text report showing the main classification metrics.

- Precision is a measure of the accuracy provided that a class label has been predicted. It is defined by: precision = TP / (TP + FP)
- Recall is true positive rate. It is defined as: Recall = TP / (TP + FN)
- F1 score: Now we are in the position to calculate the F1 scores for each label based on the precision and recall of that label. The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. It is a good way to show that a classifer has a good value for both recall and precision.

Source: Wikipedia

In this report, the precision for **'Injury Collision' Class is 0.89 (89 percent)**. It means when the model predicts 100 times 'Injury Collision', only 89 prediction is correct. But, the precision for 'Property Damage Only Collision' class is 0 (zero). It means the model always gets wrong when predicts 'Property Damage Only Collision' class. Recall for 'Injury Collision' Class is 1 (100 percent). It means when the model can predict all actual values in this class. On the other side, recall for 'Property Damage Only Collision' class is 0 (zero). It means the model can't predict actual values.

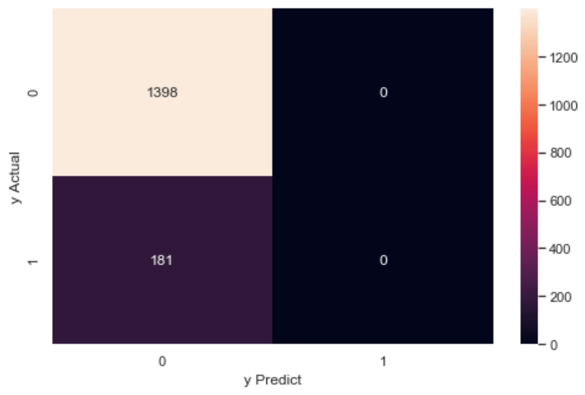


```
                                precision    recall  f1-score   support

              Injury Collision       0.89      1.00      0.94      1398
Property Damage Only Collision       0.00      0.00      0.00       181

                      accuracy                           0.89      1579
                     macro avg       0.44      0.50      0.47      1579
                  weighted avg       0.78      0.89      0.83      1579
```

Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

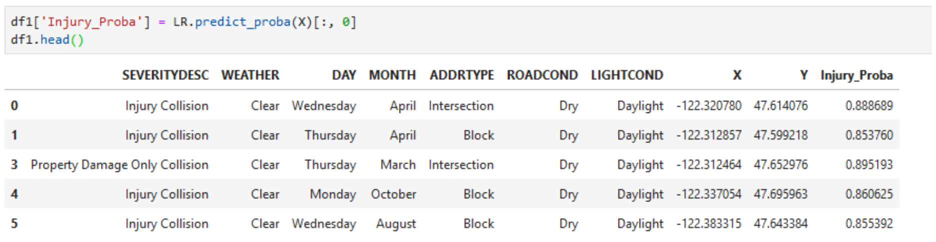
It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table). Source: Wikipedia



In the confusion matrix below, we can see a TP of 1398. It means that the entire class (0) 'Injury Collision' can be predicted entirely. However, the 181 'Property Damage Only Collision' class are predicted to be 'Injury Collision' (False Positive).

INSIGHT: Time and condition with high probability of injury collision

One of the advantages of using Logistic Regression is that we can get probabilities from predictions. This probability we can merge with the initial dataset. Thus, we can analyze **times and conditions** which have **high or low probability**. A new column is added to the dataset with the name 'Injury_Proba'.

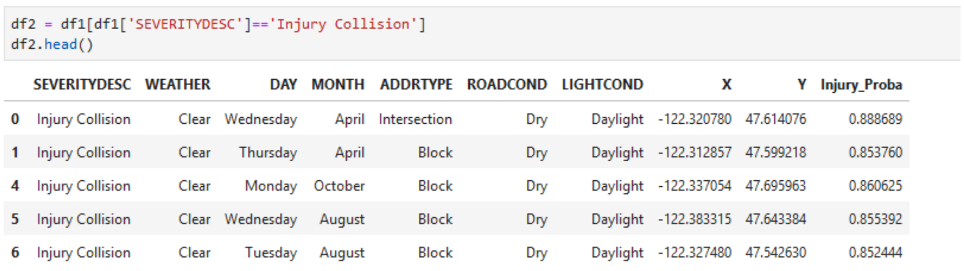


```
df1['Injury_Proba'] = LR.predict_proba(X)[:, 0]
df1.head()
```

	SEVERITYDESC	WEATHER	DAY	MONTH	ADDRTYPE	ROADCOND	LIGHTCOND	X	Y	Injury_Proba
0	Injury Collision	Clear	Wednesday	April	Intersection	Dry	Daylight	-122.320780	47.614076	0.888689
1	Injury Collision	Clear	Thursday	April	Block	Dry	Daylight	-122.312857	47.599218	0.853760
3	Property Damage Only Collision	Clear	Thursday	March	Intersection	Dry	Daylight	-122.312464	47.652976	0.895193
4	Injury Collision	Clear	Monday	October	Block	Dry	Daylight	-122.337054	47.695963	0.860625
5	Injury Collision	Clear	Wednesday	August	Block	Dry	Daylight	-122.383315	47.643384	0.855392

Because the precision and recall scores for 'Injury Collision' are very high, the filtered dataset only displays this class. Besides, the precision and recall scores for the 'Property Damage Only Collision' class are also very low or zero.

This filter will also support analyzing what time and conditions on a low and high probability of an injury collision.



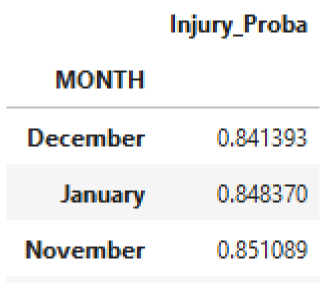
```
df2 = df1[df1['SEVERITYDESC']=='Injury Collision']
df2.head()
```

	SEVERITYDESC	WEATHER	DAY	MONTH	ADDRTYPE	ROADCOND	LIGHTCOND	X	Y	Injury_Proba
0	Injury Collision	Clear	Wednesday	April	Intersection	Dry	Daylight	-122.320780	47.614076	0.888689
1	Injury Collision	Clear	Thursday	April	Block	Dry	Daylight	-122.312857	47.599218	0.853760
4	Injury Collision	Clear	Monday	October	Block	Dry	Daylight	-122.337054	47.695963	0.860625
5	Injury Collision	Clear	Wednesday	August	Block	Dry	Daylight	-122.383315	47.643384	0.855392
6	Injury Collision	Clear	Tuesday	August	Block	Dry	Daylight	-122.327480	47.542630	0.852444

The Safest Time for Cycling Activities

What is the safest month to host a cycling event?

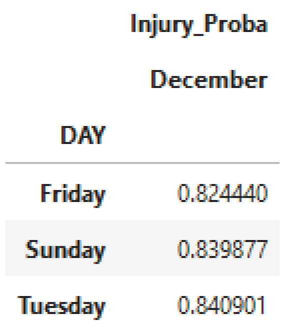
The months with the lowest ‘injury collision’ probability are **December, January, and November.** These months have a low injury collision probability.



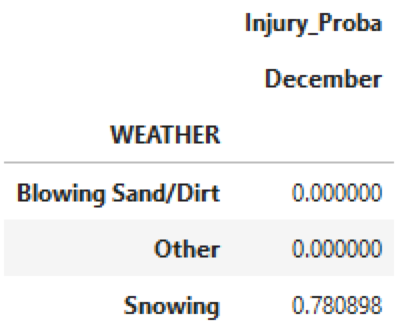
MONTH	Injury_Proba
December	0.841393
January	0.848370
November	0.851089

What is the safest day and weather to host a cycling event in December?

The days with the lowest ‘injury collision’ probability are **Friday, Sunday, and Tuesday.** The weather with the lowest ‘injury collision’ probability is **Blowing Sand/Dirt**. These times conditions have a low injury collision probability.



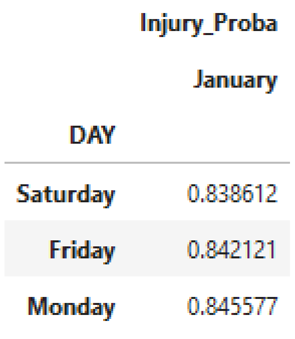
	Injury_Proba
	December
DAY	
Friday	0.824440
Sunday	0.839877
Tuesday	0.840901



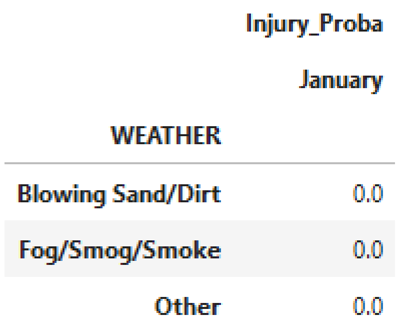
	Injury_Proba
	December
WEATHER	
Blowing Sand/Dirt	0.000000
Other	0.000000
Snowing	0.780898

What is the safest day to host a cycling event in January?

The days with the lowest ‘injury collision’ probability are **Saturday, Friday, and Monday**. The weather with the lowest ‘injury collision’ probability is **Blowing Sand/Dirt**. These times conditions have a low injury collision probability.



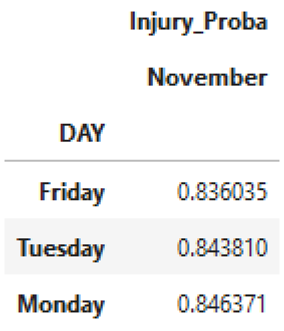
	Injury_Proba
	January
DAY	
Saturday	0.838612
Friday	0.842121
Monday	0.845577



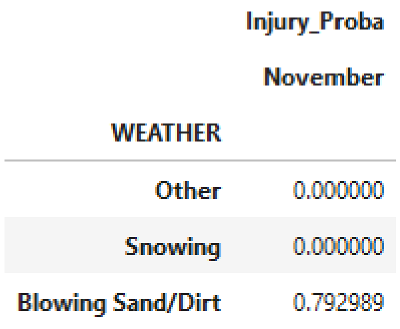
	Injury_Proba
	January
WEATHER	
Blowing Sand/Dirt	0.0
Fog/Smog/Smoke	0.0
Other	0.0

What is the safest day to host a cycling event in November?

The days with the lowest ‘injury collision’ probability are **Friday, Tuesday, and Monday**. The weather with the lowest ‘injury collision’ probability is **Snowing**. These times conditions have a low injury collision probability.



	Injury_Proba
	November
DAY	
Friday	0.836035
Tuesday	0.843810
Monday	0.846371

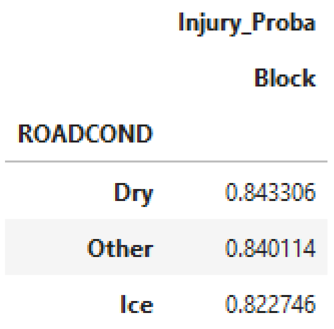


	Injury_Proba
	November
WEATHER	
Other	0.000000
Snowing	0.000000
Blowing Sand/Dirt	0.792989

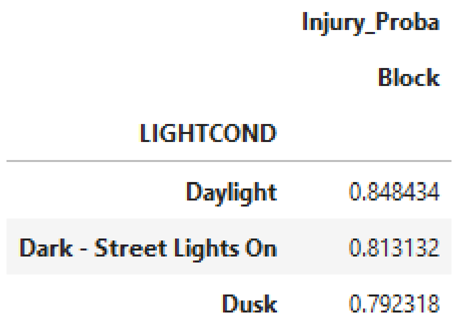
The Most Dangerous Location for Cycling Activities

What are the most dangerous light and road conditions in Block Area for cycling activities?

The road condition with the highest 'injury collision' probability in Block Area is **Dry** condition. The light condition with the highest 'injury collision' probability in Block Area is **Daylight** condition. These location conditions have a high injury collision probability.



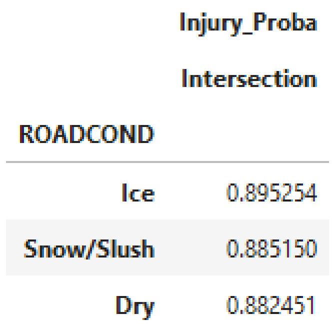
	Injury_Proba
	Block
ROADCOND	
Dry	0.843306
Other	0.840114
Ice	0.822746



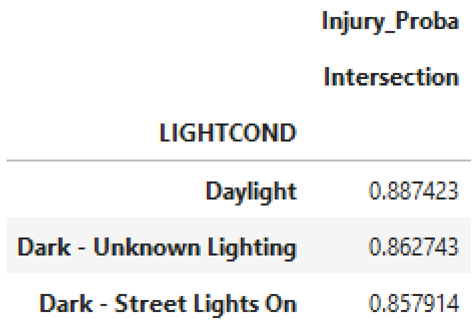
	Injury_Proba
	Block
LIGHTCOND	
Daylight	0.848434
Dark - Street Lights On	0.813132
Dusk	0.792318

What are the most dangerous light and road conditions in Intersection Area for cycling activities?

The road condition with the highest 'injury collision' probability in Intersection Area is **Ice** condition. The light condition with the highest 'injury collision' probability in Intersection Area is **Daylight** condition. These location conditions have a high injury collision probability.



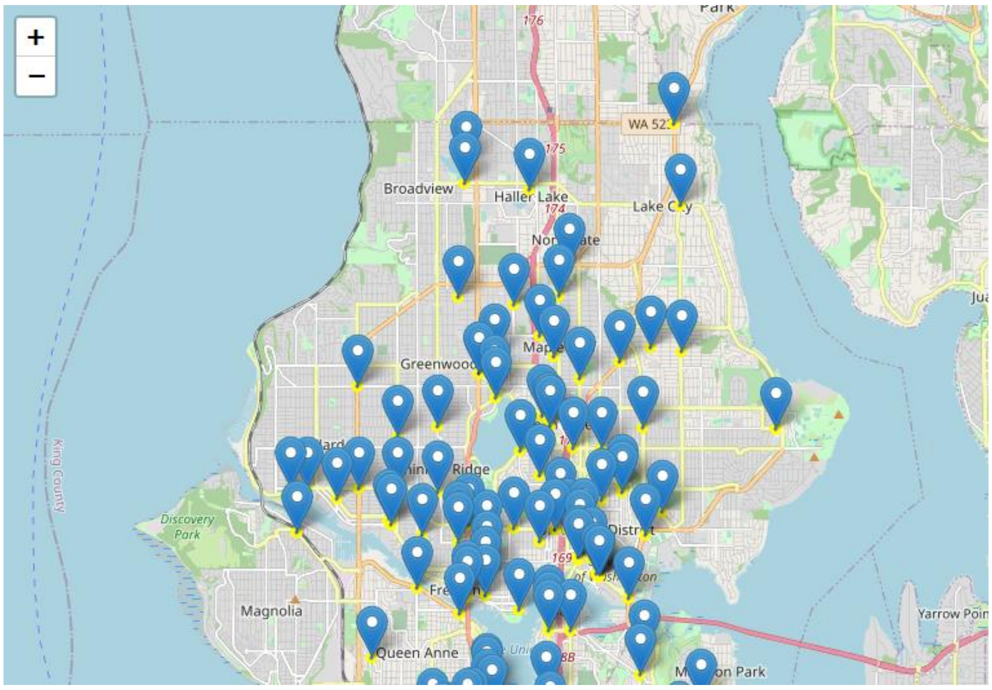
	Injury_Proba
	Intersection
ROADCOND	
Ice	0.895254
Snow/Slush	0.885150
Dry	0.882451



	Injury_Proba
	Intersection
LIGHTCOND	
Daylight	0.887423
Dark - Unknown Lighting	0.862743
Dark - Street Lights On	0.857914

Most Dangerous Locations Map for Cyclist in Seattle

The map below shows the most dangerous locations for cyclists in Seattle. The location points shown below have an 'injury collision' **probability of above 90 per cent**. Event organizers need to be careful at this location. Before the event starts, the committee needs to monitor the road conditions around that spot. If it is dangerous, the committee can repair it or give a unique sign to be more careful.



Results and Discussion

Cycling event organizers may consider the findings of this project before holding a cycling event. Organizers need to plan when the best time to maintain the event is. Besides, also be aware of points on the road in Seattle that have a high probability of injury collisions.

The safe months for hosting events for cyclists are December, January, and November. These three months have the lowest average probability of injury collision compared to other months. If the committee chooses December, then the safest choice of days is Friday, Saturday, and Tuesday. The safe weather conditions this month are blowing sand/dirt. The choice of safe days and weather in January is Saturday, Friday and Monday, with blowing sand/dirt weather conditions. This condition has the lowest average 'injury collision' probability. The following safest month option is November. This month, the committee can choose Friday, Tuesday, and Monday. However, unlike the month choices above, the safe weather conditions in November are snowing. This condition has the lowest probability of 'injury collision'.

Apart from time conditions, the committee also needs to be aware of dangerous spots for cyclists. This spot has a high probability of injury collision involving motorists and cyclists. The block area's most dangerous conditions are 'dry' road conditions and 'daylight' lighting conditions. This condition has the highest average probability of injury collision. How about the intersection area? The committee must be aware of spots with 'ice' road conditions and 'daylight' lighting conditions. Before the event starts, the committee needs to avoid routes with these conditions.

Conclusion

- The safest months to host cycling events are December, January and November. Of these three months, Friday is the safe choice. Despite the ‘daylight’ lighting conditions, the committee still needs to be careful in ‘dry’ and ‘ice’ road conditions.
- This dataset has limitations both in quality and quantity. If the class ‘property damage only collision’ is added, the precision and recall will increase. Also, if the dataset contains borough or district features, we can create a choropleth map.
- Event organizers can collaborate with the Seattle Department of Transportation (SDOT) to ensure safe cycling events with the best choice of cycling times and routes.