San Jose Traffic Accidents Prediction

Ping Chen, Mandy Wong, Nihanjali Mallavarapu and Dhruwaksh Dave

Abstract—Traffic Accidents are everywhere and happened every time all over the world. According to the statistics showing in the Association for Safe International Road Travel, nearly 1.25 million people die in road crashes each year and 20-50 million people are injured or disabled. There are so many factors to make this happen. The Bad weather conditions, for example, rain, snow, ice, and windy, etc, is one of the major factors and will be increasing the possibility of a road crash than sunny days. This project will be focusing on weather-related factors and the traffic volume to predict the possibility of traffic accidents will occur in San Jose, CA. In this project, we analyze the relationship between weather conditions and the number of car accidents and the relationship between the traffic volume, road network and the number of car accidents. We choose four algorithms to train the data and the accuracy for each model is around 80%.

1 Introduction

Road Safety is an important issue nowadays and also recognized as a major public health concern. It is a shared responsibility for all of us. When traffic accidents have happened around you, it would ruin your day. We all like to avoid or prevent any kind of traffic accidents happened. Apart from having a good driving attitude and fastening your seatbelt while you are driving. It is also better to know if you are in good condition to drive in different weather conditions to prevent any traffic accidents happened. Traffic accidents are a serious problem. It is not just about car damage, but could also lead to injury or even death. On the other hand, the weather is a natural condition. Even now that we are more precise about predicting the weather, we still cannot predict what will happen on the street. Therefore, the weather condition is still one of the major factors of traffic accidents. As a human, we cannot control the weather, but we can control our behavior. We are hoping this project could help to reduce traffic accidents in any weather conditions, through to minimize the major uncontrollable factor. Also, traffic accidents are not just affecting the party, it is also affecting others behind the accident. Every traffic accident is causing a certain level of traffic jams. As a student commuting to San Jose State University for classes, we always spare more time more than the GPS estimated time to arrive. We all have wasted so much time stuck in a traffic jam on the way to school. When this project helped to reduce the accidents, we are also hoping it could help save us time for commuting.

2 Brief Literature Survey

There are several papers have demonstrated the methods to predict traffic accident . Some of the papers we reviewed are outlined briefly below:

In [1], the authors collected huge heterogeneous urban datasets to predict whether an accident will occur or not for each road segment in each hour. The datasets included all the motor vehicle crashes in the state of Iowa from 2006 to 2013, detailed road network, and hourly weather data such as rainfall, temperature, census data which gives the population corresponding to a sub-area. Then they preprocessed the datasets by interpolating the missing values in weather related features and matched crash data with road networks, etc. They compared four classification models: linear SVM, Decision Tree, Random Forest and Deep Neural Networks. Based on the datasets which includes 415,000 crashes containing 40 features, the results showed that DNN got the highest accuracy 0.9512.

In [2], the authors using two supervised learning models(ANN and Decision Tree) to predict traffic accidents. They divided the features into four key factors: driver factors, road factors, vehicle factors and climate factors. To reduce the complexity of the model, they did dimension reduction based on domain knowledge and other techniques. Then they split the datasets into training set and test sets, using ANN and Decision Trees to build the model. The experiment conducted on 4861 crash records and 14 attributes. The accuracy of ANN model was 79.8% and accuracy of Decision Tree is 77.7%.

Chang, et al employed a negative binomial regression model and an ANN model to analyze accident data for National Freeway 1 in Taiwan. They investigated the relationship between vehicle accidents and highway geometry, traffic characteristics and environment conditions. The number of sections used for model estimation is 1500, and the number of sections used for testing is 492. For the negative binomial regression model, the overall model prediction accuracy for the training data is about 58.3%, while that for the testing data is about 60.8%. For the ANN model, the overall model prediction performances for the training data and the testing data are 64% and 61.4%, respectively. The author concluded that ANN is a consistent alternative for analyzing freeway accident frequency by comparing the prediction performance with negative binomial regression analysis.

[4] presented a two steps methods to prediction roadway traffic crash. The SSM(state-space model) was developed in the first step to identify the dynamic evolution process of the roadway systems that are caused by the changes of traffic flow and predict the changes of impact factors in roadway systems. Using the predicted impact factors, the SVR(support vector regression) model was incorporated in the second step to perform the traffic crash prediction. This model was evaluated in a five-year dataset that obtained from 1152 roadway segments. The proposed models result in an average prediction MAPE of 7.59%, a MAE of 0.11, and an RMSD of 0.32.

3 Methodology

This section presents how we will conduct the experiment design to predict San Jose traffic accident based on historical data. First we will briefly introduce data preparation, then we will talk about four machine learning algorithms to conduct the experiment design. At last, we will present how we verify the results of each model.

3.1 Data preparation

Motor Vehicle Crash Data: We obtained crash data in the City of San Jose DOT[5]. This data set shows the location of individual crashes where one or more fatalities and/or severe injuries occurred during the five-year period of 2013 to 2017. The data including 940 crash data. Since these crashes are mapped to the nearest intersections, each crash contains the following information: Crash Location/Intersection, Date/Time, Injured Party, and Number of Fatal or Severe Injuries per crash.

Road Networks: We collected road datasets from San Jose DOT with basic information in San Jose, street name, nearest intersection, speed limits, street segment length and the most recent average daily traffic.

Climate Data: We also obtained the historical weather data in the website of California Agriculture & Natural Resources [6]. The weather information we retrieved including observation time, precipitation amount, max temperature and min temperature.

3.2 Data preprocessing

After we got this data, we will evaluate the data quality of the dataset we acquired. We plan to perform below steps in data preprocessing [7]:

Data Cleaning: Since dirty data can cause confusion for the mining procedures for dealing with incomplete or noisy data, they are not always robust [7]. Instead, they may concentrate on avoiding overfitting the data to the function being modeled. Therefore, it is essential to run the data through some data cleaning routing. Clean the data by filling in missing values, smoothing noisy data, identifying or removing outliers and resolving inconsistencies. For example, in our scenario, the road network features contain many missing values (eg. the average daily traffic data is not available for some roads). Therefore, we need to fill in the missing values either by using a global constant or using mean/median, etc.

Data Integration: Merge multiple datasets into a coherent data store. This can improve the accuracy and speed of the subsequent data mining process. When matching attributes from one database to another during integration, special attention must be paid to the structure of the data. And we also need to do correlation analysis in order to avoid redundancy and data value conflict detection to avoid different attribute values from different sources.

Data reduction: This techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. Mining on the reduced data set should be more efficient yet produce the same analytical results. Data reduction strategies include dimensionality reduction, numerously reduction and data compression. We can pick up one or two methods to process our data.

3.3 Algorithms

In this project, we need to find the machine learning algorithm which can get the highest accuracy for classification. We are going to talk about four machine learning models, points out the difference and find the most suitable model for solving our problem.

Logistic Regression: Logistic Regression is commonly used to estimate the probability that an instance belongs to a particular class. If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled "1"), or else it predicts that it does not (i.e., it belongs to the negative class, labeled "0"). This makes it a binary classifier. Logistic Regression model computes a weighted sum of the input features (plus a bias term), and it outputs the logistic of this result [8]. We need to train the model to find the best weights in order to get the smallest cost.

Support Vector Machines: SVM is another algorithm for classification by finding a hyperplane in an N-dimensional Space to classify the data points. The hyperplane is selected to find the maximum distance between the classes [8]. The hyperplane is learned from training data using an optimization procedure that maximizes the margin. SVM has three important parameters: C, gamma and kernel. In this project, we plan to fine tune the parameters to find the most desirable outcome of this model.

Random forests: A random forest multi-way classifier consists of a number of trees, with each tree grown using some form of randomization. The leaf nodes of each tree are labeled by estimates of the posterior distribution over the image classes. Each internal node contains a test that best splits the space of data to be classified [8].

K-Nearest Neighbours: The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. KNN is a Lazy learning algorithm which simply stores training data and waits until it is given a test tuple. KNN is a completely non-parametric algorithm and it is much more efficient when the decision boundary is highly non-linear.

3.4 Evaluation

In this project, we label the dataset into two categories: 0 for non-accident and 1 for accident. Then we split 70% of total datasets to be training data and the remaining 30% to be test data. We will train our training set using the four models mentioned above then evaluate each model based on test set. Finally we will evaluate each model based on the following factors:

1. Compare the training accuracy and test accuracy, decide whether overfitting occurs or not;
2. Find the model delivers the highest accuracy.

4 Data Visualizations and Preprocessing

4.1 Weather data visualization

The weather data is specified in San Jose, CA from January 1st 2000 to October 20th 2019 including the date, weather record time, the precipitation, the highest temperature of the day, the lowest temperature of the day and the observed temperature in record time. The dataset have been cleaned by deleting the columns that all cells are null and filled in the mean value of the highest temperature and lower temperature to the missing value in observed temperature column.

The precipitation and the temperature are mainly used from this weather dataset to be considered as factors in the prediction model, which is combining with the date of the traffic accidents happened in the crash dataset and finding the correlation between the weather and the possibility of traffic accidents happened in the prediction model .

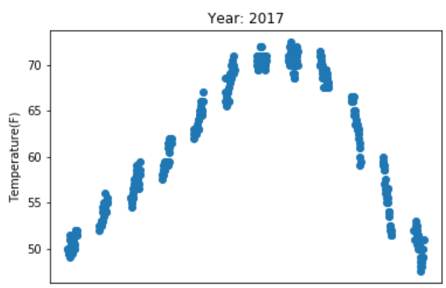

Fig. 1. Scatter plot showing the weather pattern

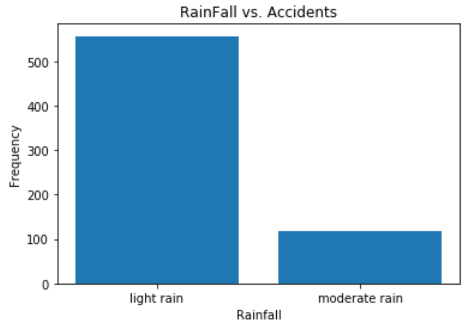

Fig. 2. Bar plot showing the Rainfall correlated to the number of traffic accidents

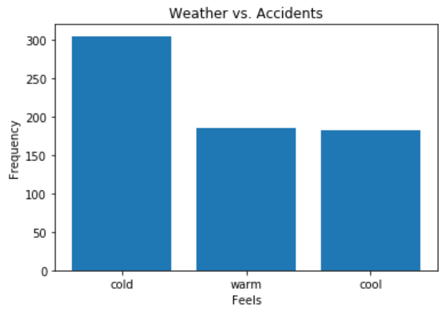

Fig. 3. Bar plot showing the relationship between the weather and the number of traffic accidents

According to the graphs showing above,traffic accidents are more likely to happen when the weather is cold and/or there is light rain. Based on the weather pattern, fall and winter will possible be cool or cold weather and/or moderate or light rain. There is shower of rain in San Jose sometimes in summer time.
Other than the cold weather, 2 traffic accidents are more likely to happen when the weather is warm and more than 2 traffic accidents are more likely to happen when the weather is cool. When the rainfall level is moderate rain, it is also possible to have a traffic accident occurs in San Jose.
To conclude, the traffic accidents are most likely to happen in San Jose when the weather is cold and/or rainfall level is light rain in Fall and Winter.

4.2 Traffic Volume Data Visualization

Create a scatter plot to visualize traffic volume with geographical information. The pattern looks exactly like San Jose. I used a color map which ranges from blue (low values) to red(high values). The high-density areas are approximately around downtown, which has a lot of intersections. The pointer with lighter colors are located on the main road or near the highway entrance.
A histogram below shows the number of instances that have a traffic volume at given value range. The max traffic volumes is 58274.0 and min value is 100, average value is 12365.

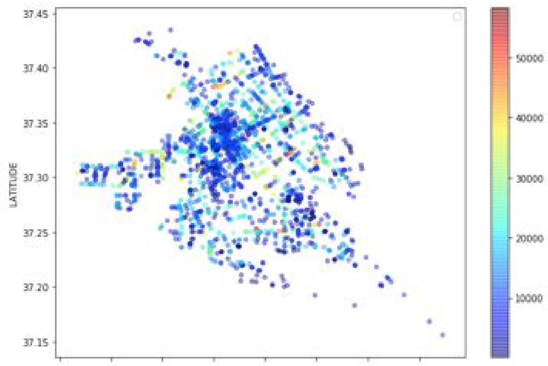

Fig.4 A geographical scatter plot of ADT

Arrange the data by traffic volume we can see the top 10 items. It is reasonable that the main road like Tully RD, Capitol Express and Blossom Hill RD have the highest daily traffic volume.

```
traffic.sort_values(by=['ADT'],ascending = False).head(10)
```

Out[311]:

	LATITUDE	LONGITUDE	ADT	STREETONE	DIRECTION	STREETTWO	NEARINTERS
221	37.315811	-121.835566	58274.0	Tully Rd	E of	McLaughlin Av	Tully Rd & McLaughlin Av
1336	37.355506	-121.834558	57107.0	CAPITOL EXPY	E of	MASSAR	CAPITOL EXPY&MASSAR
276	37.320453	-121.828685	56344.0	Tully Rd	E of	Alvin Av	Tully Rd & Alvin Av
401	37.257227	-121.797167	56000.0	Blossom Hill Rd	E of	Hwy-101	Blossom Hill Rd & Hwy-101
219	37.323364	-121.945314	51883.0	Stevens Creek Bl	E of	Baywood Av	Stevens Creek Bl & Baywood Av
973	37.322281	-121.826050	51822.0	Tully Rd	W of	King Rd	Tully Rd & King Rd
1029	37.382344	-121.900446	49988.0	BROKAW RD	E of	RIDDER PARK DR	BROKAW RD & RIDDER PARK DR
971	37.314997	-121.836772	49545.0	Tully Rd	W of	McLaughlin Av	Tully Rd & McLaughlin Av
280	37.313068	-121.839772	48908.0	Tully Rd	E of	Lucretia Av	Tully Rd & Lucretia Av
951	37.365579	-121.849631	48289.0	McKee Rd	E of	Jackson Av	McKee Rd & Jackson Av

Fig. 5 Top 10 items ranging by traffic volume

4.3 Road Network Data Visualization

In speed survey data, it lacks the important location values about latitude and longitude of the intersection in each item, which is useful in data integration with crash data. I used googlemap geocode API to find the latitude and longitude for each item. Plot the boxplot of latitude and attitude to make sure that the API return the correct values. There is one outlier for latitude and 6 outliers for longitude. I have checked that the corresponding items located in the boundaries of San Jose and the latitude and longitude values are correct.

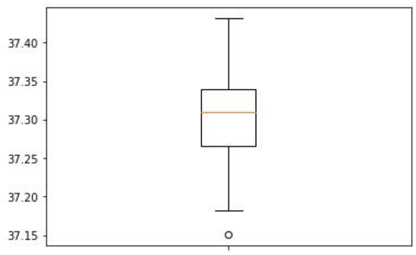

Fig. 6 Latitude box plot

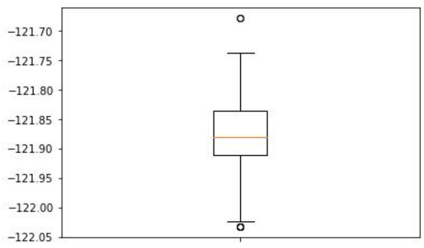

Fig.7 Longitude box plot

Create a scatter plot to visualize speed limit with geographical information and a histogram to see the number of instances located in a given range.

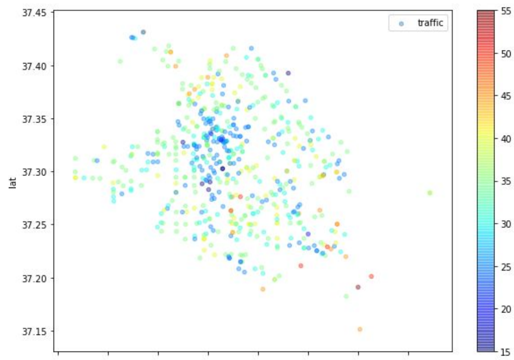

Fig. 8 A geographical scatter plot of Speed limit

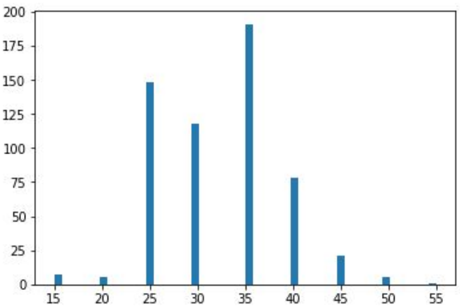

Fig. 9 Histogram of speed limit

In order to gain insight into the data, I created a correlation matrix and a correlation heatmap about the features we are interested: speed limit, the number of accident records, road length, speed limit, average speed, 85th speed. We can see from Fig.10 that all these features have a positive correlation coefficients.

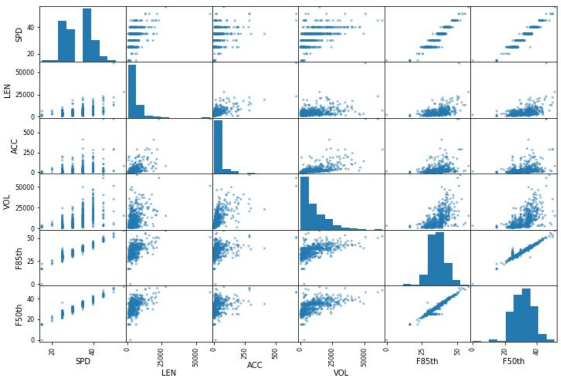

Fig. 10 Correlation matrix

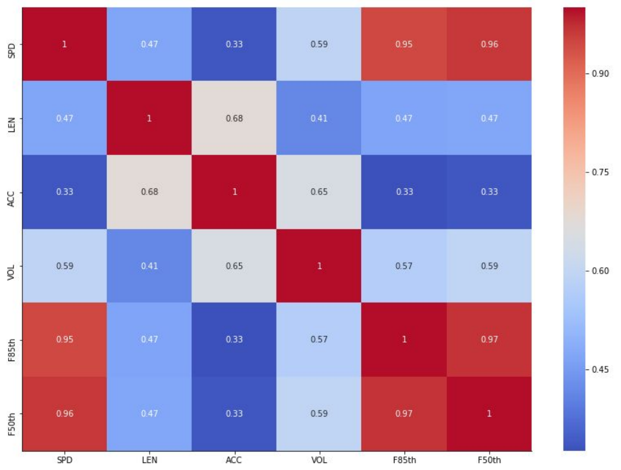

Fig. 11 Correlation Heatmap

4.4 Handle missing data:

After getting insight about speed survey data and there are some missing data in these datasets. For example, features of F85th, F50, road length, ACC(the number of historical traffic accident). The F85th means the speed at or below which 85 percent of all vehicles are observed to travel under free-flowing conditions past a monitored point and the F50 is 50 percent. So I fill in F85th NA with 1.1 times of corresponding limited speed and F50 with the limited speed, which seems more reasonable according to the driving style in San Jose. There are 6 missing values in ACC, fill in with the value of the nearest intersection. For the missing traffic volume, fill in with the value of the nearest intersection by computing the Euclidean distance. In weather data: delete the columns that all cells are null, use the mean of high temp and low temp to fill in obs temp.

4.5 Data Integration:

Since we need to predict San Jose Traffic accident based on the weather condition of the accident date, the traffic volume and road information of the place that accident took, so we need to combine the four datasets we found together.

Based on the location of each item in crash data, we find the corresponding average traffic volume of that location in traffic volume data. In crash data, the accident location is mapped to the nearest intersection, like ALMADEN EX & CHERRY AV. There is one problem that the intersection name in the first csv is upper case while in the other csv, intersection name both have upper case and lower case, so we change all the intersection name to upper case for the convenience to do entity identification. The second issue is in crash csv, intersection name could be ROAD_A & ROAD_B, but in the traffic csv, intersection name is ROAD_B & ROAD_A. However, these two are the same entities from different data sources. The third problem is not all the locations in crash data can find the corresponding traffic volume in the traffic data. In this case, we use the value from its nearest intersection by computing the Euclidean Distance instead. Similarly, we combine the crash data with road network data. For the missing road information of some crash locations, we use the information of its nearest intersection instead.

After the data of Traffic volume, Road network and traffic accident have been integrated, the last step is to combine all data with the weather condition.

Data has been integrated based on the accident date. The precipitation and the temperature are mainly used from this weather dataset to be considered as factors in this prediction model to find the correlation between the weather and the possibility of traffic accidents happened.

In this project, we construct the class label based on accident(1) vs non-accident(0). Since all the data we get from the crash data are labelled 1, we need to create dataset for label 0. Because all traffic besides the crash data can be considered as non-accident, we randomly generate time and location of one incident and fetch the corresponding traffic volume, road network info and weather data.

Finally, we have total 4352 data including 852 accident data and 3500 non-accident data. The dataset seems imbalance but I think it makes sense since the accident has a lower possibility than non-accident. And we have features like 'LONGITUDE', 'LATITUDE', 'SPD', 'LEN', 'ACC', 'VOL', 'LOCAL',, 'Precip', 'Air max', 'min', 'obs'. And we split the dataset into 70% training data and 30% test data.

5 Algorithms Implementation

K Nearest Neighbor:

Choose default metrics (Euclidean metrics). I try different values of k in order to achieve the highest accuracy. Use 5-folds cross validation to acquire more accurate score. It can be seen from the plot that after k increasing to 25, accuracy will not increase. So in this dataset, k=25 achieve the highest accuracy, which is 80.39.

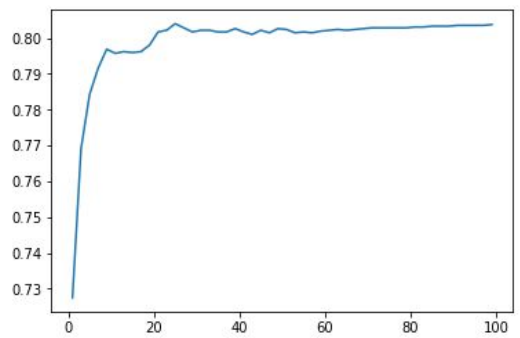

Fig. 12 Relationship between k and accuracy

Support Vector Machines:

From the implementation of Support Vector Machine, the model can predict successful with accuracy around 80%. Since the dataset is big, it took some time to build the model, but it doesn't overfit.

Logistic Regression:

From the implementation of the Logistic Regression algorithm, it can be observed that the model can predict with an accuracy of 81%. The support and confidence of the model is also high.

Random Forests:
When Random Forests classifier was applied to the data, an accuracy of 80% was obtained. Normalization of data further did not result in any significant improvements.

6 Conclusion

In this project, we collect San Jose Motor Vehicle Crash data, road network data, traffic volume data and weather data to create a whole dataset. We use this integrated dataset to predict the traffic accident. We train four models: KNN, SVM, Random Forest, Logistic Regression. All these models get an accuracy around 80%.

References

[1] Yuan, Zhuoning, et al., "Predicting traffic accidents through heterogeneous urban data: A case study." 6th International Workshop on Urban Computing (UrbComp 2017). 2017.

[2] Roop Kumar R, et al. “DATA ANALYSIS IN ROAD ACCIDENTS USING ANN AND DECISION TREE.” International Journal of Civil Engineering and Technology (IJCIET). 2018

[3] Chang, Li-Yen. "Analysis of freeway accident frequencies: negative binomial regression versus artificial neural network." Safety science 43.8 (2005): 541-557

[4] Dong, Chunjiao, et al. "Roadway traffic crash prediction using a state-space model based support vector regression approach." PloS one 14.4 (2019): e0214866.

[5] http://gisdata-csjdotgis.opendata.arcgis.com/?geometry=-123.635%2C37.161%2C-121.523%2C37.489

[6] http://ipm.ucanr.edu/WEATHER/wxactstnames.html

[7] Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011

[8] Géron, Aurélien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2017

APPENDIX

Algorithms Implementation:

```
#KNN - KNN.ipynb

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import math
import operator
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

#load final_data.csv
data = pd.read_csv('final_data.csv')
train_set, test_set = train_test_split(data, test_size = 0.2,
random_state = 42)
X_train = train_set.drop(['Accident'], axis=1)
Y_train = train_set['Accident']
X_test = test_set.drop(['Accident'], axis=1)
Y_test = test_set['Accident']
Y = data['Accident']
X = data.drop(['Accident'], axis = 1)

Y_train.value_counts()
Y_test.value_counts()

# calculate euclidean distance
def euclideaDistance(in1, in2, length1, length2):
    distance = 0
    for i in (length1, length2):
        distance += pow((in1[i] - in2[i]),2)
    return math.sqrt(distance)

# check
data.iloc[1]
len(data.iloc[1])

#K-NN Implementation
classifier = KNeighborsClassifier(n_neighbors=20)
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(Y_test, Y_pred))
print(classification_report(Y_test, Y_pred))
print("Accuracy={:.4f}".format(accuracy_score(Y_test, Y_pred)))


# accuracy
k = []
accuracy = []
for i in range(1, 50, 2):
    k.append(i)
    classifier = KNeighborsClassifier(n_neighbors=i)
    classifier.fit(X_train, Y_train)
    Y_pred = classifier.predict(X_test)
    accuracy.append(accuracy_score(Y_test, Y_pred))


plt.plot(k, accuracy)
plt.show()

max(accuracy)
k[pd.Series(accuracy).idxmax()]

k = []
accuracy = []
for i in range(1, 100, 2):
    k.append(i)
    classifier = KNeighborsClassifier(n_neighbors=i)
    scores = cross_val_score(classifier, X, Y, cv=5, scoring='accuracy')
    accuracy.append(scores.mean())

plt.plot(k, accuracy)
plt.show()

max(accuracy)
k[pd.Series(accuracy).idxmax()]


# SVM - ML_Implementation.ipynb
import pandas as pd
from sklearn.model_selection import train_test_split

data = pd.read_csv('final_data.csv')
data.drop(columns = 'Unnamed: 0', inplace = True)
print(data)

# split the data
X = data.drop('Accident',axis=1)
y = data['Accident']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)

#build SVM model and fit the model
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear',C=2,gamma=0.01)
svclassifier.fit(X_train, y_train)

#accuracy
print(svclassifier.score(X_test,y_test))

from sklearn.metrics import accuracy_score
y_pred = svclassifier.predict(X_test)
print(accuracy_score(y_test,y_pred))


# Logistics Regression - LogRegWmet.ipynb
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sns

#load data
df = pd.read_csv('accident.csv')

feature_cols = ['SPD', 'ACC', 'Precip', 'Air max','min','obs']
X = df[feature_cols] # Features
y = df.Accident # Target variable

# split X and y into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,rand
om_state=101)

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
```

```
logreg.fit(X_train,y_train)

y_pred=logreg.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

# visualization
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix),annot=True,
cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

# accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))


# RandomForests - RandomForests.ipynb
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.ensemble import RandomForestClassifier

#load the data
data = pd.read_csv('final_data.csv')

for cols in data.columns:
    print(cols)

data.drop('Unnamed: 0', inplace=True, axis=1)

corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')

corr_target = abs(corr["Accident"])
relevant_features = corr_target[corr_target>0]

for cols in data.columns:
    if cols not in relevant_features:
        print(cols)
        data.drop(f'{cols}', axis=1, inplace=True)

# split the data
X = data.drop('Accident', axis=1)
y = data['Accident']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# build and fit decision tree model
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

predictions = dtree.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))

lst = []
for val in y_test:
    lst.append(val)
score = 0
for i in range(len(predictions)):
    if predictions[i] == lst[i]:
        score += 1

print(score/len(predictions)*100)

def predict(data):
    X = data.drop('Accident', axis=1)
    y = data['Accident']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    dtree = DecisionTreeClassifier()
    dtree.fit(X_train, y_train)
    predictions = dtree.predict(X_test)
    print(confusion_matrix(y_test,predictions))
    print(classification_report(y_test,predictions))
    lst = []
    for val in y_test:
        lst.append(val)
    score = 0
    for i in range(len(predictions)):
        if predictions[i] == lst[i]:
            score += 1
    print(score/len(predictions)*100)

predict(data)

for col in data.columns:
    if col != 'Accident':
        data[col] = data[col]/data[col].mean()

predict(data)

# build and fit random forest model
clf=RandomForestClassifier(n_estimators=100,max_depth=2,random
_state=0)
clf.fit(X_train, y_train)

def predict2(data):
    X = data.drop('Accident', axis=1)
    y = data['Accident']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    clf = RandomForestClassifier(n_estimators=100,
max_depth=3,random_state=0)
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    print(confusion_matrix(y_test,predictions))
    print(classification_report(y_test,predictions))
    lst = []
    for val in y_test:
        lst.append(val)
    score = 0
    for i in range(len(predictions)):
        if predictions[i] == lst[i]:
            score += 1
    print(score/len(predictions)*100)


predict2(data)
```

Data Preprocessing:

```
# data_preprocessing.ipynb
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import array
import requests
import googlemaps
import os

speed = pd.read_csv("Speed_Pub.csv")
speed.keys()

speed = speed.drop(["OBJECTID","ROUTE","DATE","on_hold",
"CD", "PD", "GlobalID","reason_for_on_hold", "Comment",
"GlobalID", "Jurisdiction", "NEW",
"MEAN_SPD","BGN_PACE","PRCNT_PACE","Shape__Length","RA
TE","TYPE"],axis=1)
speed.head(10)

speed.keys()
speed.sort_values('VOL', ascending= False).iloc[:10,:]

gmaps =
googlemaps.Client(key='AIzaSyAbDfjZYGZhSPC60ZxAfYEm1yKxV
77zQfw')

speed['lat'] = ""
speed['lng'] = ""

speed['intersection1'] = speed['STREET'] + ' & ' + speed['START'] +',
San Jose'
speed['intersection2'] = speed['START']  + ' & ' + speed['STREET'] +',
San Jose'
speed['intersection3'] = speed['STREET'] + ' & ' + speed['END_'] +',
San Jose'
speed['intersection4'] = speed['END_'] + ' & ' + speed['STREET'] +',
San Jose'

a = 0
for i in range(0,speed.shape[0]):
    item = speed.iloc[i]
    address0 = item.intersection1
    address1 = item.intersection3
    geocode_result0 = gmaps.geocode(address0)
    geocode_result1 = gmaps.geocode(address1)
    if len(geocode_result0) != 0:
        speed.at[i, 'lat'] =
geocode_result0[0]["geometry"]["location"]["lat"]
        speed.at[i, 'lng']  =
geocode_result0[0]["geometry"]["location"]["lng"]
        a = a + 1
    elif len(geocode_result1) != 0:
        speed.at[i, 'lat'] =
geocode_result1[0]["geometry"]["location"]["lat"]
        speed.at[i, 'lng'] =
geocode_result1[0]["geometry"]["location"]["lng"]
        a = a + 1

speed.head(10)
speed.info()

speed['lat'] = speed.lat.convert_objects(convert_numeric=True)
speed['lng'] = speed.lng.convert_objects(convert_numeric=True)
speed.info()

value85  = 1.1* speed['SPD']
value50 = speed['SPD']
speed['F85th'] = value85.where(speed['F85th'] == np.nan,
speed['F85th'])
speed['F50th'] = value50.where(speed['F50th'] == np.nan,
speed['F50th'])

speed[speed.isnull().any(axis=1)].head()

speed['F85th'].fillna(1.1*speed['SPD'], inplace = True)
speed['F50th'].fillna(speed['SPD'], inplace = True)
speed['SPD']
speed.info()

plt.boxplot(speed["lat"])
quartile_1, quartile_3 = np.percentile(speed["lat"], [25, 75])
print(min(speed["lat"]))
iqr = quartile_3 - quartile_1
lower_bound = quartile_1 - (iqr * 1.5)
upper_bound = quartile_3 + (iqr * 1.5)
np.where((speed["lat"] > upper_bound) | (speed["lat"] <
lower_bound))

plt.boxplot(speed["lng"])
quartile_1, quartile_3 = np.percentile(speed["lng"], [25, 75])
iqr = quartile_3 - quartile_1
lower_bound = quartile_1 - (iqr * 1.5)
upper_bound = quartile_3 + (iqr * 1.5)
np.where((speed["lng"] > upper_bound) | (speed["lng"] <
lower_bound))

speed = speed.dropna()
speed.info()

speed.plot.scatter(x = "lng", y = "lat", alpha = 0.4, c = speed["SPD"],
label = "traffic", cmap=plt.get_cmap("jet"), figsize = (10, 7))
plt.legend()
plt.show()

speed.head(10)
plt.hist(speed.SPD, bins=50)
plt.show()

speed.to_csv(r'speed_done.csv')

accident = pd.read_csv("crash.csv")
accident.shape

accident =
accident.drop(["AccidentId","ESRI_OID","GlobalID","Fatal_MajorInj
uries"],axis=1)
accident = accident.rename(columns={'X': 'LONGITUDE', 'Y':
'LATITUDE','AccidentDateTime':'Date','AStreetNameAndSuffix':'ASt
reet','BStreetNameAndSuffix':'BStreet'})

for i in range(0,10):
    date = accident.iloc[i].Date
    accident.at[i, 'Date'] = date[0:10].replace('-','')
accident.head(10)

accident["Involving"].value_counts()

accident = accident.replace("Motorist","0")
accident = accident.replace("Pedestrian","1")
accident = accident.replace("Bicyclist","2")
accident["Involving"].value_counts()

dataset = pd.DataFrame(columns = ['LONGITUDE', 'LATITUDE',
'Date', 'AStreet', 'BStreet', 'FatalInjuries',
       'MajorInjuries', 'Involving', 'Nearest_Intersection', 'SPD', 'LEN',
'ACC', 'VOL', 'F85th', 'LOCAL',
       'F50th'])
num = 0
#for i in range(0,accident.shape[0]):
for i in range(0,accident.shape[0]):
```

```
    A = accident.iloc[i].Nearest_Intersection
    B = accident.iloc[i].AStreet
    C = accident.iloc[i].BStreet
    lat = accident.iloc[i].LATITUDE
    lng = accident.iloc[i].LONGITUDE
    i1 = speed.loc[speed["intersection1"].isin([A]) |
speed["intersection2"].isin([A]) | speed["intersection3"].isin([A]) |
speed["intersection4"].isin([A])]
    i2 = speed.loc[speed["STREET"].isin([B]) |
speed["START"].isin([B]) | speed["END_"].isin([B])]
    i3 = speed.loc[speed["STREET"].isin([C]) |
speed["START"].isin([C]) | speed["END_"].isin([C])]
    a = accident.iloc[i][0:9]
    if i1.shape[0] != 0:
        b = i1.iloc[0][3:10]
        c = pd.concat([a, b], sort=False,)
        num = num + 1
    elif i2.shape[0] != 0:
        arr = []
        for n in range(0, i2.shape[0]):
            lat1 = i2.iloc[n].lat
            lng1 = i2.iloc[n].lng
            dis = abs(lat - lat1) + abs(lng - lng1)
            arr.append(dis)
        arr.index(min(arr))
        b = i2.iloc[arr.index(min(arr))][3:10]
        c = pd.concat([a, b], sort=False,)
        num = num + 1
    elif i3.shape[0] != 0:
        arr = []
        for m in range(0, i3.shape[0]):
            lat2 = i3.iloc[m].lat
            lng2 = i3.iloc[m].lng
            dis = abs(lat - lat2) + abs(lng - lng2)
            arr.append(dis)
        arr.index(min(arr))
        b = i3.iloc[arr.index(min(arr))][3:10]
        c = pd.concat([a, b], sort=False,)
    else: c = a
    dataset = dataset.append(c, ignore_index=True)

dataset.info()

dataset = dataset.dropna()
dataset.info()
dataset.shape

dataset.head(10)

dataset.to_csv(r'accident_speed.csv')

# WeatherDataCleaning.ipynb
import pandas as pd
import matplotlib.pyplot as plt

weather = pd.read_csv('weather.csv')

weather = weather.drop(columns = 'Station')
weather = weather.dropna(axis = 'columns', how = 'all')

avg = (weather['Air max'] + weather['min'])/2
weather['obs'] = weather['obs'].fillna(avg)

print(weather.head())

Yr2013_Date = []
Yr2013_temp = []

for x in range(4749,5114):
    Yr2013_Date.append(weather['Date'][x])
    Yr2013_temp.append(weather['obs'][x])

Yr2013 = {'Date': Yr2013_Date, 'Temperature': Yr2013_temp}
Yr2013 = pd.DataFrame(Yr2013)
print(Yr2013)

plt.scatter(Yr2013['Date'], Yr2013['Temperature'])
plt.title('Year: 2013')
plt.ylabel('Temperature(F)')
plt.xticks([])
plt.show()

Yr2014_Date = []
Yr2014_temp = []

for x in range(5114,5479):
    Yr2014_Date.append(weather['Date'][x])
    Yr2014_temp.append(weather['obs'][x])

Yr2014 = {'Date': Yr2014_Date, 'Temperature': Yr2014_temp}
Yr2014 = pd.DataFrame(Yr2014)
print(Yr2014)

plt.scatter(Yr2014['Date'], Yr2014['Temperature'])
plt.title('Year: 2014')
plt.ylabel('Temperature(F)')
plt.xticks([])
plt.show()

Yr2015_Date = []
Yr2015_temp = []

for x in range(5479,5844):
    Yr2015_Date.append(weather['Date'][x])
    Yr2015_temp.append(weather['obs'][x])

Yr2015 = {'Date': Yr2015_Date, 'Temperature': Yr2015_temp}
Yr2015 = pd.DataFrame(Yr2015)
print(Yr2015)

plt.scatter(Yr2015['Date'], Yr2015['Temperature'])
plt.title('Year: 2015')
plt.ylabel('Temperature(F)')
plt.xticks([])
plt.show()

Yr2016_Date = []
Yr2016_temp = []

for x in range(5844,6210):
    Yr2016_Date.append(weather['Date'][x])
    Yr2016_temp.append(weather['obs'][x])

Yr2016 = {'Date': Yr2016_Date, 'Temperature': Yr2016_temp}
Yr2016 = pd.DataFrame(Yr2016)
print(Yr2016)

plt.scatter(Yr2016['Date'], Yr2016['Temperature'])
plt.title('Year: 2016')
plt.ylabel('Temperature(F)')
plt.xticks([])
plt.show()

Yr2017_Date = []
Yr2017_temp = []

for x in range(6210,6575):
    Yr2017_Date.append(weather['Date'][x])
    Yr2017_temp.append(weather['obs'][x])

Yr2017 = {'Date': Yr2017_Date, 'Temperature': Yr2017_temp}
Yr2017 = pd.DataFrame(Yr2017)
```

```
print(Yr2017)

plt.scatter(Yr2017['Date'], Yr2017['Temperature'])
plt.title('Year: 2017')
plt.ylabel('Temperature(F)')
plt.xticks([])
plt.show()

Yr2018_Date = []
Yr2018_temp = []

for x in range(6575,6940):
   Yr2018_Date.append(weather['Date'][x])
   Yr2018_temp.append(weather['obs'][x])

Yr2018 = {'Date': Yr2018_Date, 'Temperature': Yr2018_temp}
Yr2018 = pd.DataFrame(Yr2018)
print(Yr2018)

plt.scatter(Yr2018['Date'], Yr2018['Temperature'])
plt.title('Year: 2018')
plt.ylabel('Temperature(F)')
plt.xticks([])
plt.show()

Yr2019_Date = []
Yr2019_temp = []

for x in range(6940,7233):
   Yr2019_Date.append(weather['Date'][x])
   Yr2019_temp.append(weather['obs'][x])

Yr2019 = {'Date': Yr2019_Date, 'Temperature': Yr2019_temp}
Yr2019 = pd.DataFrame(Yr2019)
print(Yr2019)

plt.scatter(Yr2019['Date'], Yr2019['Temperature'])
plt.title('Year: 2019')
plt.ylabel('Temperature(F)')
plt.xticks([])
plt.show()

weather.to_csv('weather_clean.csv')

# Data_integration.ipynb
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.utils import shuffle

weather = pd.read_csv('weather_clean.csv')
acc_speed = pd.read_csv('accident_speed.csv')

print(acc_speed.head(20))
print(weather.head())

date_split = acc_speed['Date'].str.split("T",n=1,expand=True)
acc_speed['Accident_Date']= date_split[0]
acc_speed['Accident_Time']= date_split[1]

acc_speed = acc_speed.drop(columns='Date')
acc_speed =
acc_speed[['LONGITUDE','LATITUDE','Accident_Date','Accident_Ti
me','AStreet','BStreet','FatalInjuries','MajorInjuries','Involving','Neare
st_Intersection','SPD','LEN','ACC','VOL','F85th','LOCAL','F50th']]

acc_speed['Accident_Time'] =
acc_speed['Accident_Time'].fillna("unknown")
print(acc_speed.head())

for x in range(10):
   date = acc_speed['Accident_Date'][x]
   datetimeobject = datetime.strptime(date,'%Y%m%d')
   newformat = datetimeobject.strftime('%Y-%m-%d')
   acc_speed['Accident_Date'][x] = newformat

print(acc_speed.head(20))

for y in range(len(weather['Date'])):
   wea = str(int(weather['Date'][y]))
   datetimeobject = datetime.strptime(wea,'%Y%m%d')
   newformat = datetimeobject.strftime('%Y-%m-%d')
   weather['Date'][y] = newformat

print(weather)
print(weather.info())
print(acc_speed.info())

dataset = pd.merge(left=weather,right=acc_speed, left_on='Date',
right_on='Accident_Date')
print(dataset)

dataset = dataset.drop(columns=['Date','Time'])
print(dataset)

dataset =
dataset[['Accident_Date','Accident_Time','LONGITUDE','LATITUDE
','AStreet','BStreet',

'FatalInjuries','MajorInjuries','Involving','Nearest_Intersection','SPD','
LEN',
                'ACC','VOL','F85th','LOCAL','F50th','Precip','Air
max','min','obs']]
print(dataset)

dataset.to_csv('dataset.csv')

#integrate with false data
dataset_v6_temp = pd.read_csv('dataset_v6_temp.csv')
print(dataset_v6_temp)

dataset_v6 = shuffle(dataset_v6_temp[1:])
print(dataset_v6)

dataset_v6.to_csv('dataset_v6.csv')


# create_false_data.ipynb
import pandas as pd
from pandas import DataFrame
import numpy as np
import datetime
import random
from datetime import datetime
pd.set_option('display.max_columns', 500)

dataset = pd.read_csv('dataset.csv')
dataset.info()
dataset.shape
dataset.head(1)

dataset.keys()
data = pd.read_csv('speed_done.csv')
test = data.sample(n=3500, replace=True, random_state=1)
test.shape
test.head(1)

false_data = DataFrame(columns=['Accident_Date', 'Accident_Time',
'LONGITUDE', 'LATITUDE',
      'AStreet', 'BStreet', 'FatalInjuries', 'MajorInjuries', 'Involving',
      'Nearest_Intersection', 'SPD', 'LEN', 'ACC', 'VOL', 'F85th',
'LOCAL',
```

```
    'F50th', 'Accident'])
false_data.head(5)

weather = pd.read_csv('weather_clean.csv')
weather.head(5)
weather.keys()
weather = weather.drop(['Unnamed: 0'], axis=1)
weather.keys()

false_dataset = pd.merge(left=false_data,right=weather,
left_on='Accident_Date', right_on='Date')
false_dataset.info()

false_dataset.head(1)
false_dataset.keys()
false_dataset.shape

false_dataset = false_dataset.drop(columns=['Date','Time'])
false_dataset.shape

false_dataset.keys()
false_dataset.head(1)

false_dataset.shape
false_dataset.to_csv(r'false_data.csv')


# falsedata_dataset_integration.ipynb
import pandas as pd
from pandas import DataFrame
import numpy as np
import datetime
import random
from datetime import datetime
pd.set_option('display.max_columns', 500)

false_data = pd.read_csv('false_data.csv')
false_data.info()

dataset = pd.read_csv('dataset.csv')
dataset.shape
dataset['Accident'] = 1
dataset.shape
dataset.head(10)
dataset.keys()

false_data.keys()
false_data = false_data[['Unnamed: 0', 'Accident_Date',
'Accident_Time', 'LONGITUDE', 'LATITUDE',
    'AStreet', 'BStreet', 'FatalInjuries', 'MajorInjuries', 'Involving',
    'Nearest_Intersection', 'SPD', 'LEN', 'ACC', 'VOL', 'F85th',
'LOCAL',
    'F50th', 'Precip', 'Air max', 'min', 'obs', 'Accident']]
false_data.head(5)
dataset.head(5)

frame = [dataset, false_data]
final_data = pd.concat(frame)

final_data.shape
final_data = final_data.reset_index()
final_data.info()

for i in range(0, final_data.shape[0]):
  time = final_data.iloc[i]['Accident_Date'].split('-')
  year = int(time[0])
  month = int(time[1])
  day = int(time[2])
  final_data.at[i,'year'] = year
  final_data.set_value(i, 'year', year)
  final_data.at[i,'month'] = month
  final_data.at[i,'day'] = day
  hour = final_data.iloc[i]['Accident_Time']
  if (type(hour) == int) != 1:
    if (hour != "unknown"):
      final_data.at[i, 'hour'] =
int(final_data.iloc[i]['Accident_Time'][0:2])
    else:
      final_data.at[i, 'hour'] = random.randint(0,23)
  else:
    final_data.at[i, 'hour'] = hour

final_data.shape
final_data.to_csv(r'dataset_v2.csv')
final_data.keys()

final_data = final_data.drop(['Unnamed: 0', 'Accident_Date',
'Accident_Time', 'AStreet', 'BStreet', 'Nearest_Intersection'], axis=1)
final_data.keys()
final_data.shape
final_data = final_data[['index', 'LONGITUDE', 'LATITUDE',
'FatalInjuries', 'MajorInjuries',
    'Involving', 'SPD', 'LEN', 'ACC', 'VOL', 'F85th', 'LOCAL', 'F50th',
    'Precip', 'Air max', 'min', 'obs', 'year', 'month', 'day',
    'hour', 'Accident']]
final_data = final_data.drop(['FatalInjuries', 'MajorInjuries',
    'Involving'], axis=1)
final_data.to_csv(r'dataset_v4.csv')

final_data = pd.read_csv('dataset_v4.csv')
final_data = final_data.sample(frac=1)
final_data = final_data.reset_index()
final_data = final_data.drop(['level_0', 'index'], axis=1)
final_data.to_csv(r'dataset_v5.csv')
final_data.shape
final_data.head(10)

# test if the dataset is good enough
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(final_data, test_size = 0.2,
random_state = 42)

X_train = train_set.drop(['Accident'], axis=1)
Y_train = train_set['Accident']
X_test = test_set.drop(['Accident'], axis=1)
Y_test = test_set['Accident']

Y_train.value_counts()
Y_test.value_counts()

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

from sklearn.svm import SVC
param_C = 50
param_gamma = 50
classifier = SVC(C=1,gamma=1)


classifier.fit(X_train, Y_train)
expected = Y_test
predicted = classifier.predict(X_test)
print("Accuracy={:.4f}".format(accuracy_score(expected, predicted)))

from sklearn.model_selection import GridSearchCV
parameters = {'kernel':['rbf'], 'C':[1,2,5,10,20,50],
'gamma':[0.01,0.1,1,5,10,20,50]}
svm_clsf = SVC()
grid_clsf = GridSearchCV(estimator=svm_clsf,
param_grid=parameters, scoring='accuracy',n_jobs=16,verbose=10)
grid_clsf.fit(X_train, Y_train)
classifier = grid_clsf.best_estimator_
```

```
params = grid_clsf.best_params_
print("Best parameters set found on development set:")
print(params)
print(classifier)

y_true, y_pred = Y_test, grid_clsf.predict(X_test)
print(classification_report(y_true, y_pred))

results = grid_clsf.cv_results_
for mean, std, params in
zip(results["mean_test_score"],results["std_test_score"],results["para
ms"]):
   print("%0.3f (+-%0.03f) for %r" %(mean, std*2, params))
```

Data Visualization:

```
# Weather_Accidents_Visulization.ipynb

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

data = pd.read_csv('dataset.csv')

# Accident count on days
accident = {}
for row in data['Accident_Date']:
   if row in accident:
      accident[row] += 1
   else:
      accident[row] = 1

print(accident)

hap1 = []
hap2 = []
hap_more = []
for row in accident:
   if accident[row] == 1:
      hap1.append(row)
   elif accident[row] == 2:
      hap2.append(row)
   else:
      hap_more.append(row)

print("happened 1 accident a day:",len(hap1), "days")
print("happened 2 accidents a day:",len(hap2), "days")
print("happened more than 2 accidents a day:",len(hap_more),
"days")


weather_accident = pd.DataFrame(columns =
["Accident_Date","Precip","obs","rainfall","feel","count"])

# Precip and Obs dict
precip = {}
obs = {}
for i in range(len(data)):
   a = data['Accident_Date'][i]
   if a not in precip:
      precip[a] = data['Precip'][i]
   if a not in obs:
      obs[a] = data['obs'][i]

#fill in Accident_data and count
weather_accident['Accident_Date'] = accident.keys()
for i in range(len(weather_accident)):
   key = weather_accident['Accident_Date'][i]
   if accident[key]:
      weather_accident['count'][i] = accident[key]
      weather_accident['Precip'][i] = precip[key]
      weather_accident['obs'][i] = obs[key]

for i in range(len(weather_accident)):
   precip = weather_accident['Precip'][i]
   if precip > 2:
      weather_accident['rainfall'][i] = "violent rain"
   elif precip > 0.3:
      weather_accident['rainfall'][i] = "heavy rain"
   elif precip > 0.098:
      weather_accident['rainfall'][i] = "moderate rain"
   else:
      weather_accident['rainfall'][i] = "light rain"
```

```
for i in range(len(weather_accident)):
    temp = weather_accident['obs'][i]
    if temp > 122 :
        weather_accident['feel'][i] = "extremely hot"
    elif temp > 98.6:
        weather_accident['feel'][i] = "very hot"
    elif temp > 77:
        weather_accident['feel'][i] = "hot"
    elif temp > 68:
        weather_accident['feel'][i] = "warm"
    elif temp > 59:
        weather_accident['feel'][i] = "cool"
    elif temp > 32:
        weather_accident['feel'][i] = "cold"
    else:
        weather_accident['feel'][i] = "ice/freezes"

print(weather_accident)


# RainFall vs. Accidents
fig, ax = plt.subplots()
# count the occurrence of each class
data = weather_accident['rainfall'].value_counts()
# get x and y data
points = data.index
frequency = data.values
# create bar chart
ax.bar(points, frequency)
# set title and labels
ax.set_title('RainFall vs. Accidents')
ax.set_xlabel('Rainfall')
ax.set_ylabel('Frequency')

# Weather vs. Accidents
fig, ax = plt.subplots()
# count the occurrence of each class
data = weather_accident['feel'].value_counts()
# get x and y data
points = data.index
frequency = data.values
# create bar chart
ax.bar(points, frequency)
# set title and labels
ax.set_title('Weather vs. Accidents')
ax.set_xlabel('Feels')
ax.set_ylabel('Frequency')

# Rainfall
hap1_rainfall_list = []
hap2_rainfall_list = []
a_rainfall_list = []

for i in range(len(weather_accident)):
    count =  weather_accident['count'][i]
    if count == 1:
        hap1_rainfall_list.append(weather_accident['rainfall'][i])
    elif count == 2:
        hap2_rainfall_list.append(weather_accident['rainfall'][i])
    else:
        a_rainfall_list.append(weather_accident['rainfall'][i])

sns.countplot(hap1_rainfall_list).set_title("Most likely to have 1
accident")
sns.countplot(hap2_rainfall_list).set_title("Most likely to have 2
accidents")
sns.countplot(a_rainfall_list).set_title("Most likely to have more than
2 accident")

# Weather
hap1_weather_list = []
hap2_weather_list = []
a_weather_list = []

for i in range(len(weather_accident)):
    count =  weather_accident['count'][i]
    if count == 1:
        hap1_weather_list.append(weather_accident['feel'][i])
    elif count == 2:
        hap2_weather_list.append(weather_accident['feel'][i])
    else:
        a_weather_list.append(weather_accident['feel'][i])

sns.countplot(hap1_weather_list).set_title("Most likely to have 1
accident")
sns.countplot(hap2_weather_list).set_title("Most likely to have 2
accidents")
sns.countplot(a_weather_list).set_title("Most likely to have more
than 2 accidents")

# speed_data_visulization.ipynb
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas.plotting import scatter_matrix
import seaborn as sns

speed = pd.read_csv('speed_done.csv')

speed.plot.scatter(x = "lng", y = "lat", alpha = 0.4, c = speed["ACC"],
label = "traffic", cmap=plt.get_cmap("jet"), figsize = (10, 7))
plt.legend()
plt.show()

speed.plot.scatter(x = "lng", y = "lat", alpha = 0.4, c = speed["SPD"],
label = "traffic", cmap=plt.get_cmap("jet"), figsize = (10, 7))
plt.legend()
plt.show()

speed.plot.scatter(x = "lng", y = "lat", alpha = 0.4, c = speed['VOL'],
cmap=plt.get_cmap("jet"), figsize = (10, 7))
plt.legend()
plt.show()

speed.sort_values(by=['ACC'],ascending = False).head(10)
speed.keys()

data = speed.drop(['Unnamed: 0', 'STREET', 'START', 'LOCAL',
'END_','lat', 'lng', 'intersection1', 'intersection2', 'intersection3',
'intersection4'], axis = 1)
scatter_matrix(data, figsize=(12, 8))

corr = data.corr()
cmap = sns.diverging_palette(220, 5, as_cmap=True)
g = sns.heatmap(corr, annot=True,  cmap = 'coolwarm')
sns.despine()
g.figure.set_size_inches(14,10)
plt.show()

data.keys()
data = data.drop(['LEN', 'ACC', 'VOL'], axis=1)

corr = data.corr()
cmap = sns.diverging_palette(220, 5, as_cmap=True)
g = sns.heatmap(corr, annot=True, cmap = 'coolwarm')
sns.despine()
g.figure.set_size_inches(14,10)
plt.show()
```