MCDA5570: BIG DATA

TEAM 2

Team Member	
Gnanavel, Praveen Kumar	A00443487
Kudaibergenov, Timur	A00439340
Li, Wenshuo	A00445457
Panasuriya, Chirag B.	A00442907

Contents

Introduction

This is a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to Dec 2020. These broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 3 million accident records in this dataset.

Data Feature Description

There five classes of data in the whole dataset relevant to traffic accident, which includes influence, time relevant information, geography relevant information, traffic condition information and weather condition information.
Influence: severity, influence range in time and space.
Time relevant information: the start time and end time.
Geography relevant information: the latitude and longitude, street, city and state.
Traffic condition: the traffic condition around the accident, like if there are traffic lights etc.
Weather condition: temperature, wind chill, humidity, visibility etc.

Technologies used

We are in need to use different Spark and Visualization services to find the insight from the data as well as make predictions.

- SparkML
- Tableau
- Hive

Data Preprocessing

Data cleaning:

1) Data type cast:

```
data = data.withColumn("Amenity", data["Amenity"].cast("string")).withColumn("Bump", data["Bump"].cast("string"
    )).withColumn("Crossing", data["Crossing"].cast("string")).withColumn("Give_Way", data["Give_Way"].cast("string"
    )).withColumn("Junction", data["Junction"].cast("string")).withColumn("No_Exit", data["No_Exit"].cast("string"
    )).withColumn("Railway", data["Railway"].cast("string")).withColumn("Roundabout", data["Roundabout"].cast
    ("string")).withColumn("Station", data["Station"].cast("string")).withColumn("Stop", data["Stop"].cast("string"
    )).withColumn("Traffic_Calming", data["Traffic_Calming"].cast("string")).withColumn("Traffic_Signal",
    data["Traffic_Signal"].cast("string")).withColumn("Temperature(F)", data["Temperature(F)"].cast("double"
    )).withColumn("Wind_Chill(F)", data["Wind_Chill(F)"].cast("double")).withColumn("Humidity(%)", data["Humidity(%
    )"].cast("double")).withColumn("Pressure(in)", data["Pressure(in)"].cast("double")).withColumn("Visibility(mi)",
    data["Visibility(mi)"].cast("double")).withColumn("Wind_Speed(mph)", data["Wind_Speed(mph)"].cast("double"))
```

2) “nan” value replacement:

```
# Replace nan with Nulls,there are some value in dataset are nan which means data lossing
# First deal with the boolean type column
columns = data.dtypes
for cols, typ in columns:
    if typ != 'boolean':
        data = data.withColumn(cols,F.when(F.isnan(F.col(cols)),None).otherwise(F.col(cols)))
```

3) Missing value filling:

```
# Except the "Severity" column, which will be the label, other columns will be filled when occurs missing value
label = 'Severity'
string_cols =  [cols[0] for cols in data.dtypes if cols[1] == "string" ]
num_cols = [cols[0] for cols in data.dtypes if cols[1] == "int" or cols[1] == "double" ]
num_cols.remove(label)
bool_cols = [cols[0] for cols in data.dtypes if cols[1] == "boolean"]
# fill missing value
data = data.fillna("unknown",string_cols)
data = data.fillna(0,num_cols)
```

Data Insights and Analyzation

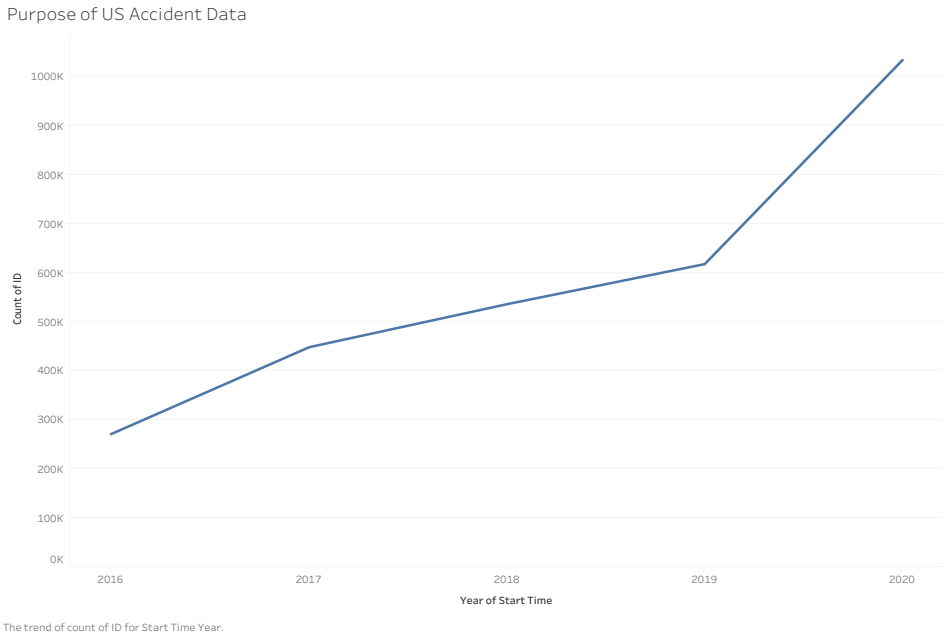


Fig 1.1

When we see the above graph, the accident rate is keep inclining without drop. It made the curiosity to learn and analyze about the problem behind the increase in accidental rate.

From the above graph, it is easy to say that accidental rate is getting increase. The next question arise in U.S which states affects the most, how and why. Let's dig further to resolve all the question which we need to know.

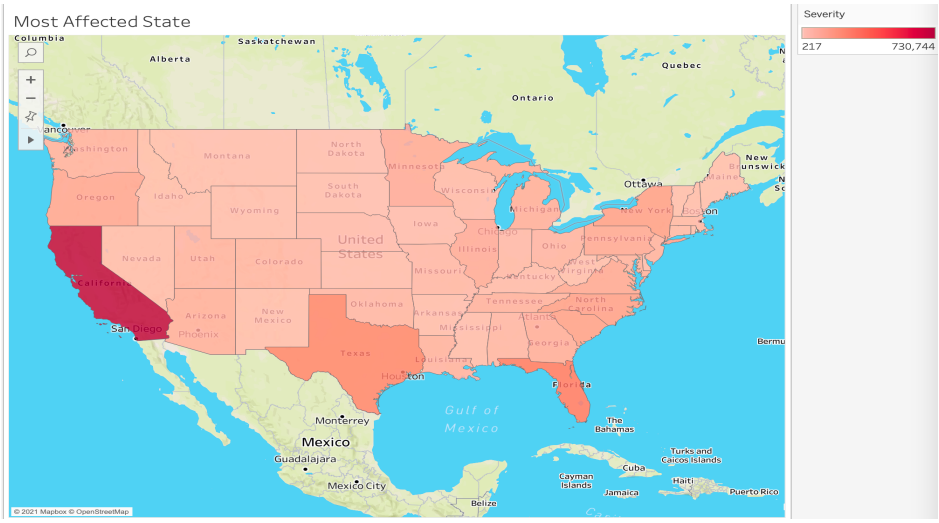


Fig 1.2

The above map shows that California is the most affected states in U.S than any other states, so it is important to start gathering the insight from California state to resolve and implement the solution in all other states.

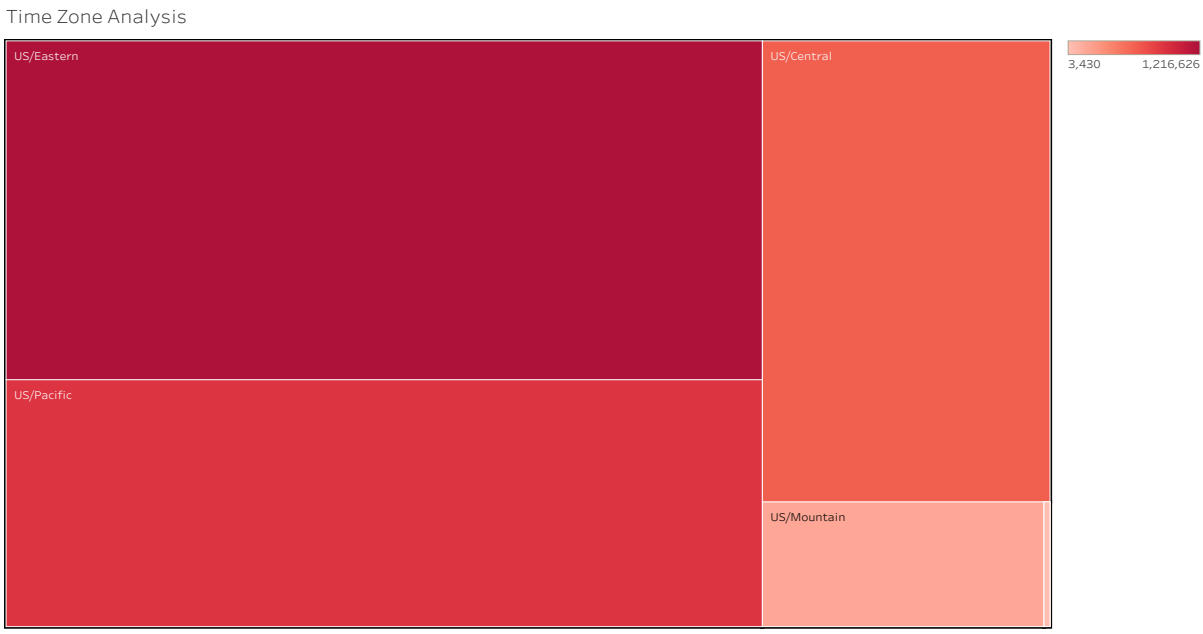


Fig 1.3

Meanwhile, the most accident rate with high severity from the Easter time zone of U.S. which should be focused on high priority.

Even the accident is high in states, the officials must focus on what time (day/night) so it will give them an idea to do some improvisation, which leads to accident rate reduction. By looking at the chart in Fig 1.4, it is crystal clear that the officials must make daytime in high priority.

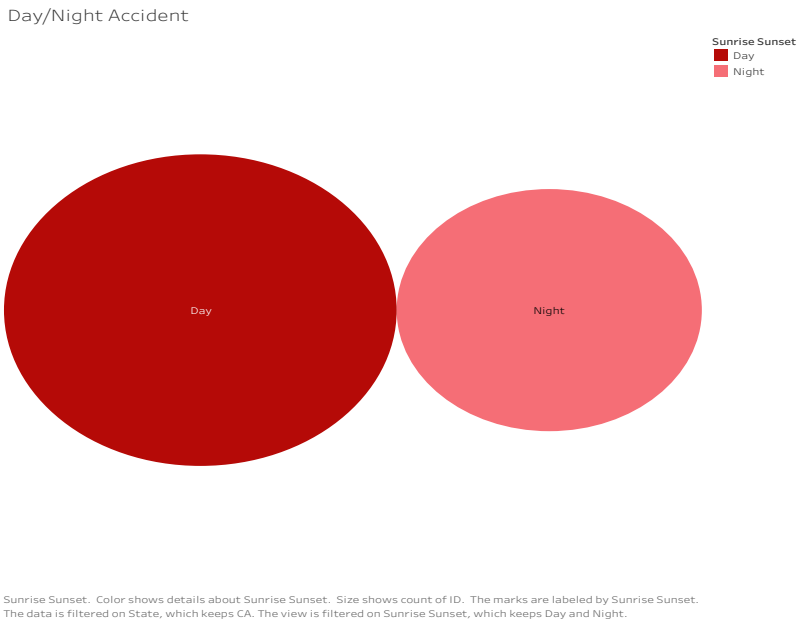


Fig 1.4

To do some improvement we have to know in daytime where and how to control accidents. The below graph will provide us the detailed insight about what are the locations it likely accidents will occur.

The below graph Fig 1.5 shows that the location where there is absence of railway, junction, station, stop or traffic signals nearby there is high chance of accidents to happen.

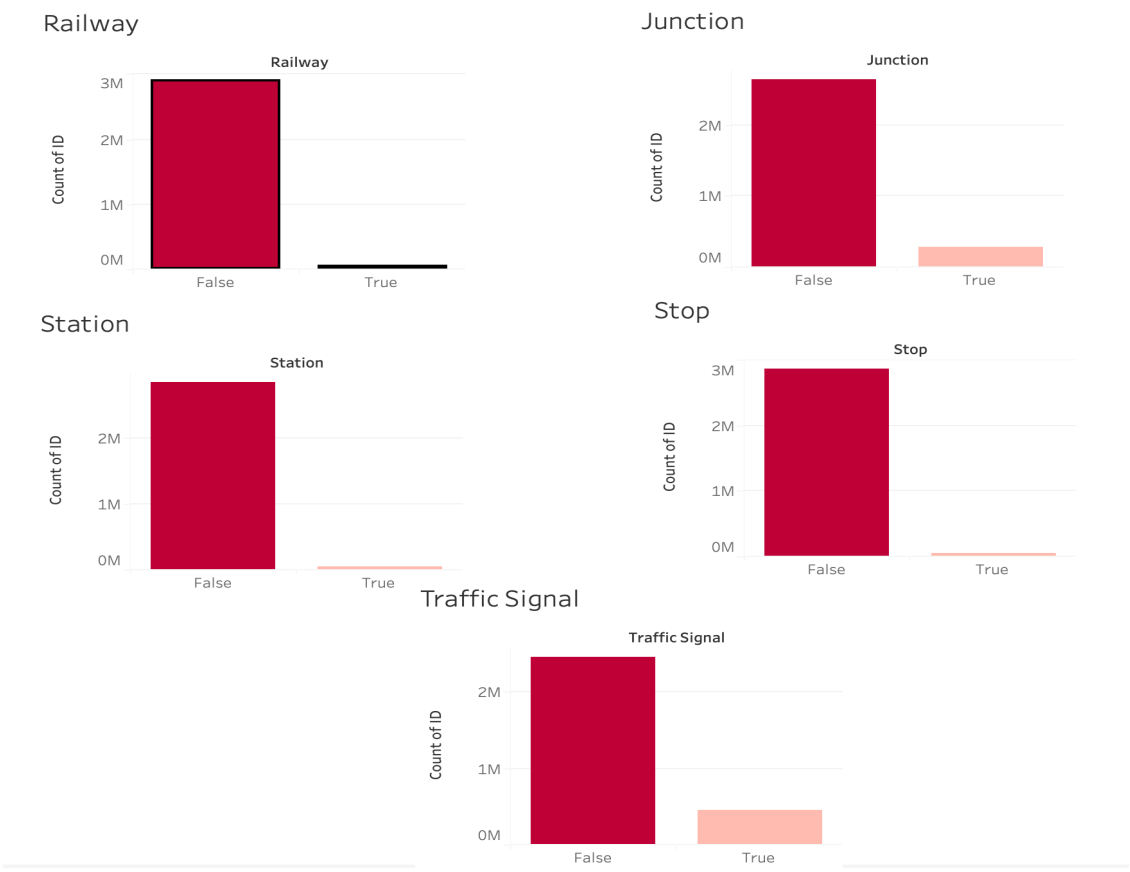


Fig 1.5

Insights Summary:

- Accident rates are inclining every year in U.S.
- Most severe case are from California (State in U.S.).
- According to time zone, Easter time zone people are most affected with high accident rates.
- Day time accidents are highly dominant than nighttime accident rates.
- Zones like absence of station, stop or railway location have high chances of accidents.
- Increasing the stop, railway, junction and traffic signal supports highly to reduce the accident rates in U.S.

Hive

Apache Hive is an open-source data warehouse software for reading, writing and managing large data set files that are stored directly in the Apache Hadoop Distributed File System (HDFS).
Hive enables SQL developers to write Hive Query Language (HQL) statements that are similar to standard SQL statements for data query and analysis.
Currently, there are about 3 million accident records in this US Accidents dataset.
First, we need to put data in HDFS as Hive works with HDFS not with our local file system.
'accidents' directory is created and put US_Accidents.csv file in HDFS with commands given in code file

```
hive> CREATE EXTERNAL TABLE us_accidents(ID STRING, Severity DECIMAL, Start_Time
 DATE, End_Time DATE, Start_Lat DECIMAL, Start_Lng DECIMAL, End_Lat DECIMAL, End
_Lng DECIMAL, Distance DECIMAL, Description STRING, Number DECIMAL, Street STRIN
G, Side STRING, City STRING, County STRING, State STRING, Zipcode STRING, Countr
y STRING, Timezone STRING, Airport_Code STRING, Weather_Timestamp STRING, Temper
ature DECIMAL, Wind_Chill DECIMAL, Humidity DECIMAL, Pressure DECIMAL, Visibilit
y DECIMAL, Wind_Direction STRING, Wind_Speed DECIMAL, Precipitation DECIMAL, Wea
ther_Condition STRING, Amenity BOOLEAN, Bump BOOLEAN, Crossing BOOLEAN, Give_Way
 BOOLEAN, Junction BOOLEAN, No_Exit BOOLEAN, Railway BOOLEAN, Roundabout BOOLEAN
, Station BOOLEAN, Stop BOOLEAN, Traffic_Calming BOOLEAN, Traffic_Signal BOOLEAN
, Turning_Loop BOOLEAN, Sunrise_Sunset STRING, Civil_Twilight STRING, Nautical_T
wilight STRING, Astronomical_Twilight STRING)
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > WITH SERDEPROPERTIES ("separatorChar" = ',')
    > LOCATION '/accidents'
    > TBLPROPERTIES('skip.header.line.count' = '1');
OK
Time taken: 0.529 seconds
```

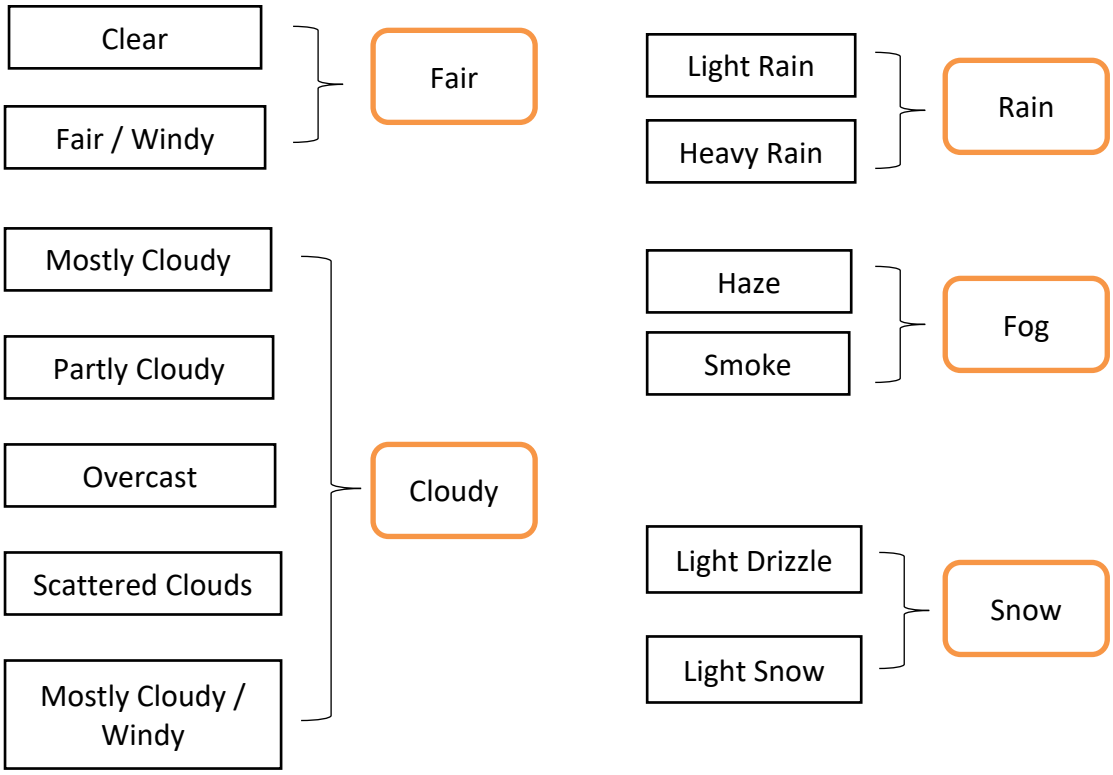
Zeppelin Notebook

Web-based notebook that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala and more.
We have used Zeppelin Notebook to fetch some markable data. Below are some queries to extract table result.
We have extracted Weekday from Start_Date to identify on which weekday most accidents happen.

```
%hive
SELECT dayOfWeek, COUNT(id) as NumberOfAccident
FROM (
    SELECT id, date_format(Start_Time ,'E') as dayOfWeek
    FROM us_accidents
    ) id
GROUP BY dayOfWeek ORDER BY NumberOfAccident DESC;
```

dayofweek	numberofaccident
Wed	507292
Fri	506750
Thu	505811
Tue	504308
Mon	475111
Sat	215791
Sun	191547

We have replaced some weather conditions with common weather name where most number of accidents happened. This helps to analyse the accidents based on weather conditions



Output is written with two new columns at the end in output.csv file.

```
hive -e """set hive.cli.print.header=true; select *, date_format
    (Start_Time ,'E') as dayOfWeek,REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
    (REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
    (REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
    (REPLACE(weather_condition,'Clear','Fair'), 'Mostly Cloudy','Cloudy'
    ),'Partly Cloudy','Cloudy'),'Overcast','Cloudy'),'Scattered Clouds'
    ,'Cloudy'), 'Light Rain', 'Rain'), 'Heavy Rain','Rain'),'Mostly Cloudy
    / Windy', 'Cloudy'),'Haze', 'Fog'),'Light Drizzle', 'Fog'),'Light
    Snow', 'Snow'),'Fair / Windy', 'Fair'),'Smoke', 'Fog'),'Cloudy /
    Windy', 'Cloudy'),'T-Storm', 'Storm'),'Light Thunderstorms and Rain',
    'Rain'),'Partly Cloudy / Windy', 'Cloudy'),'Thunder in the Vicinity',
    'Storm'),'Thunderstorm', 'Storm'),'Light Rain / Windy', 'Rain'),'Light
    Rain with Thunder', 'Rain'),'Thunder', 'Storm') as New_WeatherData
    from us_accidents"""  | sed 's/\t\+/,/g' > ~/output.csv
```

Cluster

Preparation

The clustering is performed for two types of parameters: dates and locations. The original file, US_Accidents_Dec20_Updated.csv, was downloaded and placed into the 'assignment' folder in Hadoop.

The original file has a large size (1.07GB), thus copies were created with Date and Location related columns only to improve the processing time in Zeppelin.

Analysis

To get a high-level view of locations of motor vehicle accidents (all severity), a diagram is plotted using longitude and latitude columns. The plotted diagram slightly resembles the US map due to the accidents which took place on the interstate highways and major cities. In addition to that, the plotted diagram does not render the US map accurately as it is a 2-dimensional representation, therefore it cannot depict the curvature of the earth.

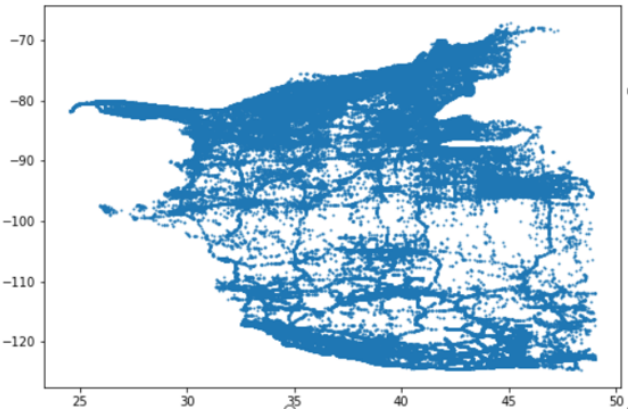
Following steps were carried out to identify the cluster centers for locations:

1. Load date into dataset
2. Create a vector assembler
3. Train a k-means model
4. Make predictions
5. Evaluate clustering using Silhouette score

```
Silhouette with squared euclidean distance = 0.8043692689013141
Cluster Centers:
[1.50706367e+09 1.50707528e+09]
[1.58418508e+09 1.58419441e+09]
```

```
%pyspark
predictions.show()
```

```
+----------+----------+--------------------+----------+
|unix_Start|  unix_End|            features|prediction|
+----------+----------+--------------------+----------+
|1558427395|1558430980|[1.558427395E9,1....|         1|
|1570470189|1570477370|[1.570470189E9,1....|         1|
|1607896380|1607899440|[1.60789638E9,1.6...|         1|
|1523983883|1523987446|[1.523983883E9,1....|         0|
|1472665249|1472667049|[1.472665249E9,1....|         0|
|1539794436|1539796218|[1.539794436E9,1....|         0|
|1576144132|1576145885|[1.576144132E9,1....|         1|
|1576972740|1576974726|[1.57697274E9,1.5...|         1|
|1527094224|1527115824|[1.527094224E9,1....|         0|
|1548837858|1548839657|[1.548837858E9,1....|         1|
|1576830012|1576836305|[1.576830012E9,1....|         1|
|1608223089|1608233813|[1.608223089E9,1....|         1|
|1524142137|1524163737|[1.524142137E9,1....|         0|
|1502904370|1502906157|[1.50290437E9,1.5...|         0|
|1590608737|1590610537|[1.590608737E9,1....|         1|
+----------+----------+--------------------+----------+
```



The process was run with multiple cluster numbers (setK(#)), however the best Silhouette score (0.864) was achieved with 2 clusters. The prediction was generated for the locations where the incidents already took place.

To see the high-level understanding of number of motor vehicle accidents in the US, a following diagram was plotted using timestamps and respective incident counts per month. Looking at the plotted diagram, we can conclude that the number of motor vehicle accidents had been increasing between February 2016 and December 2020, with the highest number of incidents in December 2020.

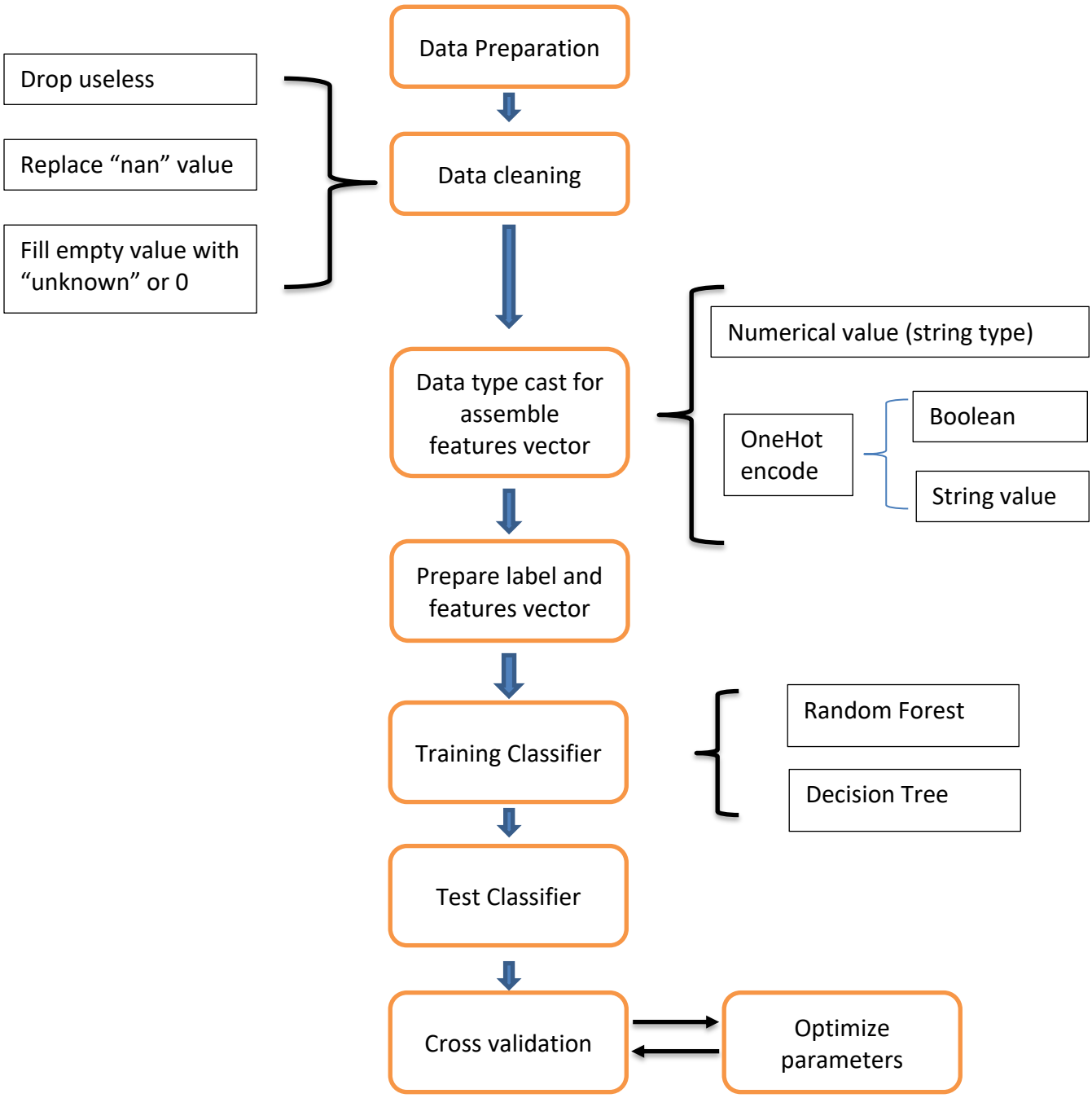
Following steps were carried out to identify the cluster centers for time:

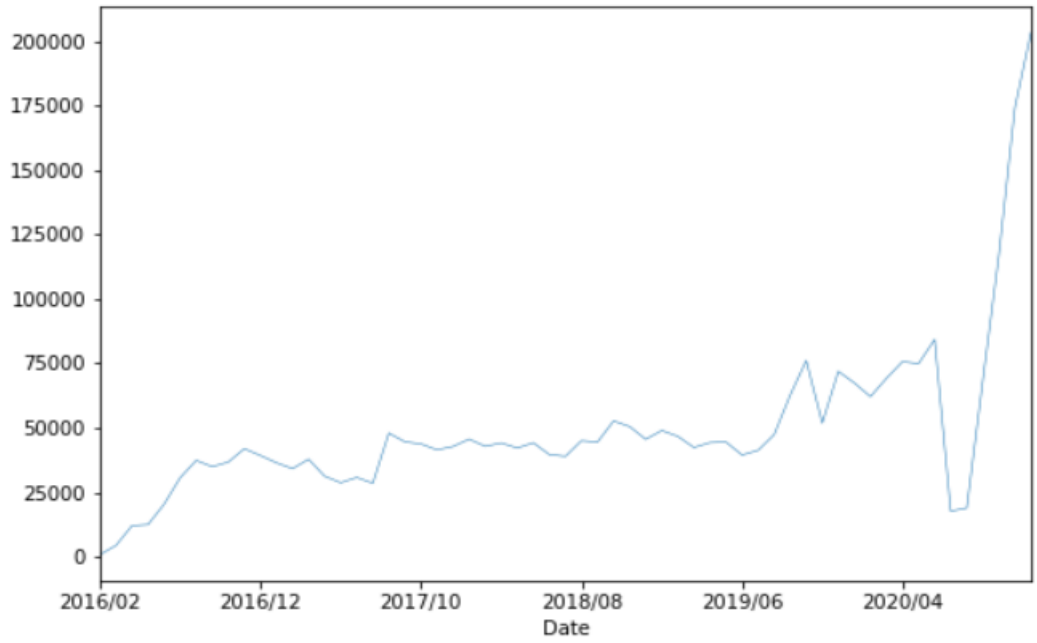
1. Load date into dataset
2. Convert from string to timestamp
3. Convert from timestamp to integer
4. Create a vector assembler
5. Train a k-means model
6. Make predictions
7. Evaluate clustering using Silhouette score

Spark Machine Learning Implement - Traffic Accident dataset classification

The main target of this part is trying to implement the machine learning technique of spark and train a classification model based on the dataset will got.

Following is the basic procedure of how we implement the classification, which is consist of four basic stages: 1) Data preparation and cleaning; 2) Lable select and features vector preparation; 3) Classifier model training; 4) Cross validation and parameter tuning.





As describe previously, the first stage is data preparation and cleaning. As the dataset we found is structured, there is no need to transform it in structure. But there are some issues on the data type and value. For example, the temperature is numerical value, its content is numerical, but data type is string. Besides, there are some missing values in dataset, which filled with string "nan".

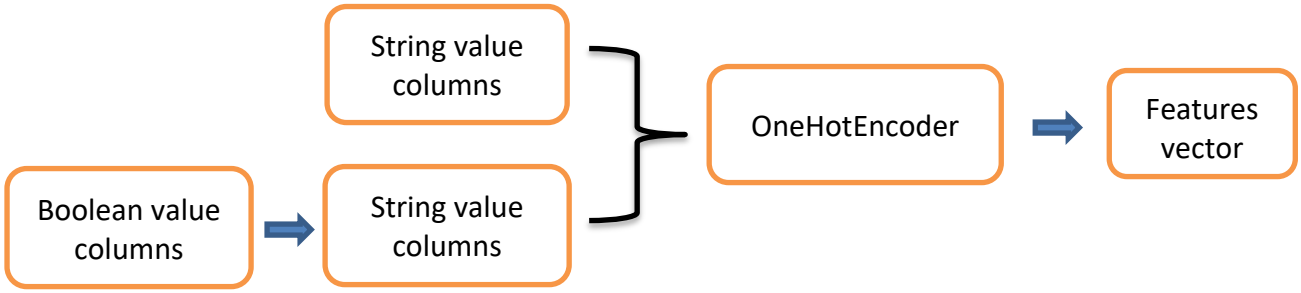
```
%pyspark
df.printSchema()
```

```
|-- Number: string (nullable = true)
|-- Street: string (nullable = true)
|-- Side: string (nullable = true)
|-- City: string (nullable = true)
|-- County: string (nullable = true)
|-- State: string (nullable = true)
|-- Zipcode: string (nullable = true)
|-- Country: string (nullable = true)
|-- Timezone: string (nullable = true)
|-- Airport_Code: string (nullable = true)
|-- Weather_Timestamp: string (nullable = true)
|-- Temperature(F): string (nullable = true)
|-- Wind_Chill(F): string (nullable = true)
|-- Humidity(%): string (nullable = true)
|-- Pressure(in): string (nullable = true)
|-- Visibility(mi): string (nullable = true)
|-- Wind_Direction: string (nullable = true)
|-- Wind_Speed(mph): string (nullable = true)
```

V	W
Temperature(F)	Wind_Chill(F)
81	81
84.6	nan
69.8	nan
73	73
88	88
44.1	42.5
72	nan
64	nan
66	66
37	32
nan	nan
30.2	18.9
46	40.6
66.9	nan
77	nan

So, first we implement the data cleaning. First, use "withColumn" and "cast" function to transform the data type. Second, replace missing value (numerical value with 0, string value with unknown).

The second stage, select label column and assemble features for model training use. The numerical value is good to use. We only need to indexing string columns and Boolean columns to vector, which is implement based on "OneHotEncoder".



The third stage, use the classifier that provided in Spark Mlib to train classification model with the prepared label and features vector.
The last stage, we use ParamGridBuilder to build parameters permutations and process cross validation.

Result: even though we try our best to optimize the parameter, but the best prediction accuracy we got is from decision tree, 62.03%