notebook

January 31, 2022

1 Reducing the number of high fatality accidents

1.1 Background

You work for the road safety team within the department of transport and are looking into how they can reduce the number of major incidents. The safety team classes major incidents as fatal accidents involving 3+ casualties. They are trying to learn more about the characteristics of these major incidents so they can brainstorm interventions that could lower the number of deaths. They have asked for your assistance with answering a number of questions.

1.2 The data

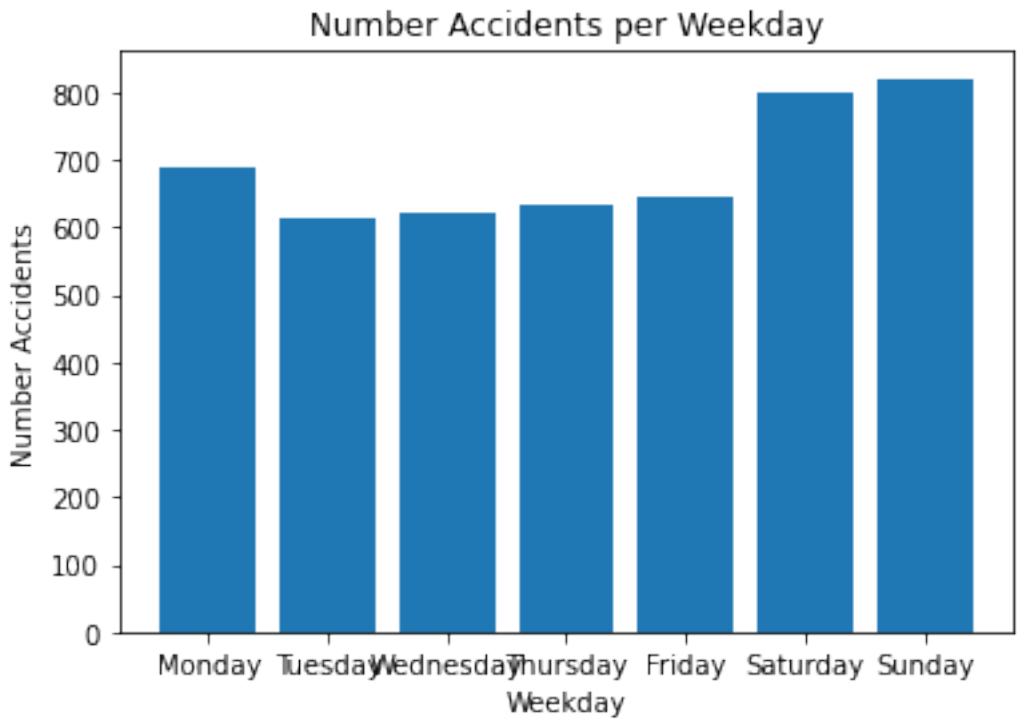
The reporting department have been collecting data on every accident that is reported. They've included this along with a lookup file for 2020's accidents.

Published by the department for transport. https://data.gov.uk/dataset/road-accidents-safety-data Contains public sector information licensed under the Open Government Licence v3.0.

[2]:
```
'''
What time of day and day of the week do most major incidents happen?
Are there any patterns in the time of day/ day of the week when major incidents␣
↪occur?
What characteristics stand out in major incidents compared with other accidents?
'''
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt

accidents = pd.read_csv(r'./data/accident-data.csv')
aDF = pd.DataFrame(accidents).dropna() #there was no empty data
#print(df.columns)

#DATE TIME DATAFRAME:
timeDF=aDF.filter(["accident_index", "number_of_casualties", "date",␣
↪"day_of_week", "time"])
timeDF=timeDF[timeDF['number_of_casualties']>2].reset_index(drop=True)

nums=[]
```

```
weekday=["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",␣
 ↪"Sunday"]
print("Calculating accident count day of week: ")
for i in range(1, 8):
    day = timeDF[timeDF['day_of_week'] == i]
    count=day['day_of_week'].count()
    nums.append(count)
    #print(count, " on ", weekday[i-1])
#print("Done counting accident count in day of week.")

#First plot:
plt.bar(weekday, nums)
plt.xlabel('Weekday')
plt.ylabel('Number Accidents')
plt.title('Number Accidents per Weekday')
plt.show()

'''
Calculating accident count day of week:
687  on  Monday
614  on  Tuesday
621  on  Wednesday
632  on  Thursday
645  on  Friday
798  on  Saturday
820  on  Sunday
Done counting accident count in day of week.
'''

'''
On what areas would you recommend the planning team focus their brainstorming␣
 ↪efforts to reduce major incidents?
'''
```

```
Calculating accident count day of week:
```

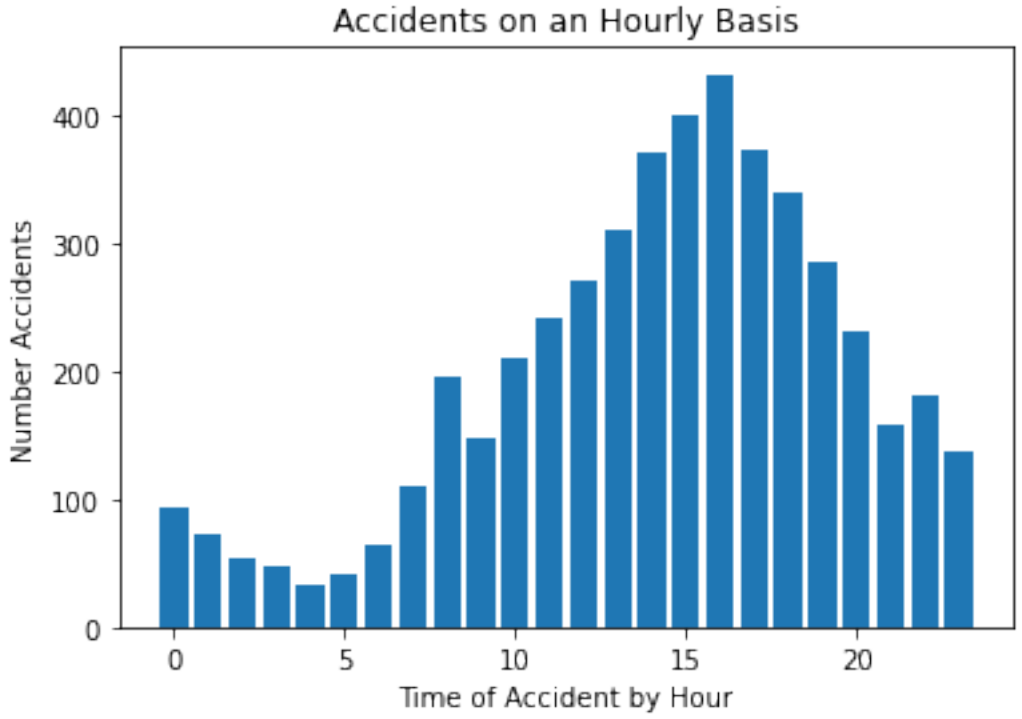


[2]:
```
'\nOn what areas would you recommend the planning team focus their brainstorming efforts to reduce major incidents?\n'
```

[3]:
```
'''
What time do most accidents happen?
'''
from datetime import datetime

hours=list(timeDF['time'].astype(str).str[:2])
hoursDic={} #store into dictionary with hours by key and number accidents as␣
 ↪values

for i in range(0,10):
    count=hours.count('0'+str(i))
    hoursDic[i]=count
for i in range(10,24):
    count=hours.count(str(i))
    hoursDic[i]=count

print(hoursDic)

plt.bar(hoursDic.keys(),hoursDic.values())
plt.title('Accidents on an Hourly Basis')
```

```
plt.xlabel('Time of Accident by Hour')
plt.ylabel('Number Accidents')
plt.show()

'''
The graph shows that most accidents happen between hours 14 and 18.
This would be between 2 and 6pm.
The most accidents happen at 4pm around when most people get off work so it is␣
↪not surprising.
'''
```

```
{0: 94, 1: 73, 2: 54, 3: 48, 4: 33, 5: 41, 6: 65, 7: 110, 8: 197, 9: 149, 10:
211, 11: 242, 12: 272, 13: 312, 14: 372, 15: 401, 16: 433, 17: 374, 18: 341, 19:
286, 20: 232, 21: 159, 22: 181, 23: 137}
```



[3]:
```
'\nThe graph shows that most accidents happen between hours 14 and 18. \nThis
would be between 2 and 6pm.\nThe most accidents happen at 4pm around when most
people get off work so it is not surprising.\n'
```

[31]:
```
'''
#LocDF will locate the most common places for accidents using longitude and␣
↪latitude.
'''
```

```
locDF=aDF.filter(['longitude', 'latitude']).round(2)
locDF = locDF.applymap(str) #need to combine two numbers in next step
locDF['Loc']=locDF['longitude']+ locDF['latitude'] #combines two dataframes as␣
 ↪str. Now, count to see if same number populates.
loc=sorted(list(locDF['Loc']))

from collections import Counter
count=Counter(loc) #counts how many in a given list
topTen=count.most_common(10) #only need the top ones

print("Top ten most dangerous locations: ")
print("Lat  " , " Long ", " Count ")
for t in topTen:
    lat=t[0][:6]
    long=t[0][6:]
    count=t[1]
    print(lat, " ", long, "  ", count)

'''
Top ten most dangerous locations:
Lat    Long   Count
-0.165   1.52    91
-0.135   1.43    75
-0.151   .5    74
-0.175   1.52    71
-0.135   1.51    70
-0.145   1.54    70
-0.065   1.51    65
-0.075   1.52    64
-0.145   1.52    64
-0.075   1.58    62
'''
```

```
Top ten most dangerous locations:
Lat    Long   Count
-0.165   1.52    91
-0.135   1.43    75
-0.151   .5    74
-0.175   1.52    71
-0.135   1.51    70
-0.145   1.54    70
-0.065   1.51    65
-0.075   1.52    64
-0.145   1.52    64
-0.075   1.58    62
```

[31]:
```
'\nTop ten most dangerous locations: \nLat    Long    Count \n-0.165    1.52
91\n-0.135    1.43    75\n-0.151    .5    74\n-0.175    1.52    71\n-0.135    1.51
70\n-0.145    1.54    70\n-0.065    1.51    65\n-0.075    1.52    64\n-0.145    1.52
64\n-0.075    1.58    62\n'
```

[4]:
```
ldata = pd.read_csv(r'./data/road-safety-lookups.csv')
#print(lDF.head()) # table field name code/format label

# Total missing values for each feature
def checkNA():
        lDF.isnull().sum()
'''
table            0
field name       0
code/format      8
label           10
note           119
'''

print("Here are the top ten accident types: ")
lDF = pd.DataFrame(data=ldata, columns=['table', 'field name'])
lcount=lDF.groupby(by=['field name']).count().sort_values(by='table',␣
 ↪ascending=False).head(10)
print(lcount)
'''
Here are the top ten accident types:
                                        table
field name
junction_detail                            11
weather_conditions                         10
carriageway_hazards                        10
special_conditions_at_site                 10
road_surface_conditions                     9
road_type                                   8
pedestrian_crossing_physical_facilities     8
day_of_week                                 7
junction_control                            7
second_road_class                           7
'''
```

```
Here are the top ten accident types:
                                        table
field name
junction_detail                            11
weather_conditions                         10
carriageway_hazards                        10
special_conditions_at_site                 10
```

```
road_surface_conditions                      9
road_type                                    8
pedestrian_crossing_physical_facilities      8
day_of_week                                  7
junction_control                             7
second_road_class                            7
```

[4]:
```
'\nfield name                        \njunction_detail
 11\nweather_conditions             10\ncarriageway_hazards
 10\nspecial_conditions_at_site     10\nroad_surface_conditions        9\n'
```

1.3 Competition challenge

Create a report that covers the following:

1. What time of day and day of the week do most major incidents happen?
2. Are there any patterns in the time of day/ day of the week when major incidents occur?
3. What characteristics stand out in major incidents compared with other accidents?
4. On what areas would you recommend the planning team focus their brainstorming efforts to reduce major incidents?

1.4 Judging criteria

CATEGORY	WEIGHTING	DETAILS

| **Recommendations** | 35% |

Clarity of recommendations - how clear and well presented the recommendation is.

Quality of recommendations - are appropriate analytical techniques used & are the conclusions valid?

Number of relevant insights found for the target audience.

|

| **Storytelling** | 30% |

How well the data and insights are connected to the recommendation.

How the narrative and whole report connects together.

Balancing making the report in depth enough but also concise.

| | **Visualizations** | 25% |

Appropriateness of visualization used.

Clarity of insight from visualization.

| | **Votes** | 10% |

Up voting - most upvoted entries get the most points.

1.5 Checklist before publishing into the competition

- Rename your workspace to make it descriptive of your work. N.B. you should leave the notebook name as notebook.ipynb.
- Remove redundant cells like the judging criteria so the workbook is focused on your story.
- Make sure the workbook reads well and explains how you found your insights.
- Check that all the cells run without error.

[ ]: