Visualization

February 5, 2021

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as pp
import pandas.plotting
#data frame is table or 2D array
#here we use df as data frame which is taking data set imported by using read function
af = pd.read_csv(r'C:\Users\Desktop\Data Science\download phn\data science data set\acci
#print df prints hole table
af.head()
print(af.shape)
print(af.columns)
print(af['Accident_Severity'].value_counts())
pd.DataFrame(af.Time.value_counts())
```

In [ ]:
```
#From describe function it is clear that Accident_Index is the primary key of the datase
#df.isnull().any() will give you boolean result about which columns contain NaN value
print(af.isnull().any())
#df.isnull().sum() gives you the total number of NaN in each column
print(af.isnull().sum())
```

In [ ]:
```
#.dropna() function drop the NaN values from the row
af.dropna(subset = ["LSOA_of_Accident_Location"],axis=0,inplace=True)
print(af.isnull().sum())
```

In [ ]:
```
x = af[af== -1].any()
print(x)
y= af[af==-1].sum()
print(y)
```

In [ ]:
```
# since we have colum which have -1 values will change it to nan
p = -1
#   .replace() will replace the value from -1 to nan
af["Junction_Control"].replace(p , np.nan , inplace=True)
af["2nd_Road_Class"].replace(p , np.nan , inplace=True)
af["2nd_Road_Number"].replace(p , np.nan , inplace=True)
af["Road_Surface_Conditions"].replace(p , np.nan , inplace=True)
af["Did_Police_Officer_Attend_Scene_of_Accident"].replace(p , np.nan , inplace=True)
```

```
        af["2nd_Road_Number"].fillna(af["2nd_Road_Number"].mode()[0],inplace=True)
        af["Road_Surface_Conditions"].fillna(af["Road_Surface_Conditions"].mode()[0],inplace=Tru
        af["Did_Police_Officer_Attend_Scene_of_Accident"].fillna(af["Did_Police_Officer_Attend_S
        x = af[af== -1].any()
        print(x)
        y= af[af==-1].sum()
        print(y)
        afnew= af.drop(['Junction_Control','2nd_Road_Class'],axis=1)
        print(afnew.dtypes)
        print(afnew.shape)
```

```
In [ ]: afjoin1 = afnew[['Accident_Severity',
               'Number_of_Vehicles', 'Number_of_Casualties', 'Day_of_Week',
                'Local_Authority_(District)','1st_Road_Class', '1st_Road_Number', 'Road_Type', '
               'Junction_Detail', '2nd_Road_Number',
               'Pedestrian_Crossing-Human_Control',
               'Pedestrian_Crossing-Physical_Facilities', 'Light_Conditions',
               'Weather_Conditions', 'Road_Surface_Conditions',
               'Special_Conditions_at_Site', 'Carriageway_Hazards',
               'Urban_or_Rural_Area', 'Did_Police_Officer_Attend_Scene_of_Accident']]
        print(afjoin1['Accident_Severity'].value_counts())

        #afjoin1.to_csv(r'C:\Users\Desktop\traffic uk\accidentjoin1.csv',index=None,header=True)
```

```
In [ ]: import pandas as pd
        import numpy as np
        import matplotlib
        import matplotlib.pyplot as plt
        import seaborn as sns
        from scipy import stats
        #data frame is table or 2D array
        #here we use df as data frame which is taking data set imported by using read function
        af = pd.read_csv(r'C:\Users\Prakhar Geete\Desktop\traffic uk\accidentjoin1.csv')
        #print df prints hole table
        af.head()
        print(af.shape)
        print(af.columns)
        sns.boxplot(x="Accident_Severity",y="Speed_limit",data=af)
        print(af['Accident_Severity'].value_counts())
```

```
In [ ]: afjoin2 = af[['Accident_Severity',
               'Number_of_Vehicles', 'Number_of_Casualties', 'Day_of_Week',
                'Local_Authority_(District)','1st_Road_Class', '1st_Road_Number', 'Road_Type', '
               'Junction_Detail',
               'Pedestrian_Crossing-Human_Control',
               'Pedestrian_Crossing-Physical_Facilities', 'Light_Conditions',
               'Weather_Conditions',
```

```
        'Urban_or_Rural_Area']]
afjoin2.to_csv(r'C:\Users\Desktop\traffic uk\accidentjoin3.csv',index=None,header=True)
```

In [ ]:
```
afdayofweek= af.groupby('Accident_Severity').Day_of_Week.value_counts()
afdayofweek
afjunction= af.groupby('Accident_Severity').Junction_Detail.value_counts(normalize=True)
afjunction
afweather= af.groupby('Accident_Severity').Weather_Conditions.value_counts(normalize=Tru
afweather
aflight= af.groupby('Accident_Severity').Light_Conditions.value_counts(normalize=True)
aflight
afroadtype=af.groupby('Accident_Severity').Road_Type.value_counts(normalize=True)
afroadtype
afurbanrural=af.groupby('Accident_Severity').Urban_or_Rural_Area.value_counts(normalize=
afurbanrural
afroadsurface=af.groupby('Accident_Severity').Road_Surface_Conditions.value_counts(norma
afroadsurface
```

In [ ]:
```
plt.figure(figsize=(5,5))
af.Accident_Severity.value_counts().plot(kind = 'bar',color=['blue','red','orange']);plt
""" 1= Fatal  2= Severe  3= Slight """
```

In [ ]:
```
plt.figure(figsize=(5,5))
af.Accident_Severity.value_counts().plot(kind = 'barh',color=['blue','red','orange']);pl
""" 1= Fatal  2= Severe  3= Slight """
```

In [ ]:
```
plt.figure(figsize=(5,5))
af.Accident_Severity.value_counts().plot(kind = 'pie',colors=['blue','red','orange']);pl
""" 1= Fatal  2= Severe  3= Slight """
```

In [ ]:
```
plt.figure(figsize=(10,10))
afdayofweek.plot(kind='barh')
print("1= Fatal  2= Severe  3= Slight")
print("\n1: Monday  2: Tuesday  3: Wednesday  4: Thursday  5: Friday  6: Saturday  7: Su
```

In [ ]:
```
plt.figure(figsize=(10,10))
afdayofweek.unstack().plot(kind='bar');plt.title('Accident_severit, Days_of_week')
print("Accident Severity 1= Fatal  2= Severe  3= Slight")
print("""\nDays_of_week
1: Monday
2: Tuesday
3: Wednesday
4: Thursday
5: Friday
6: Saturday
7: Sunday""")
```

In [ ]:
```
plt.figure(figsize=(10,10))
afdayofweek.unstack().plot(kind='bar',stacked = True)
print("1= Fatal  2= Severe  3= Slight")
print("\n1: Monday  2: Tuesday  3: Wednesday  4: Thursday  5: Friday  6: Saturday  7: Su
```

```
In [ ]: plt.figure(figsize=(10,10))
        afjunction.unstack().plot(kind='bar');plt.title('Accident_severit, Junction_Details')
        print("Accident Severity: 1= Fatal  2= Severe  3= Slight")
        print("""\nJunction_Details
        0 :Not at junction or within 20 metres
        1: Roundabout
        2: Mini-roundabout
        3: T or staggered junction
        5:Slip road
        6:Crossroads
        7:More than 4 arms (not roundabout)
        8:Private drive or entrance
        9:Other junction""")

        #afjunction.plot(kind='barh')

In [ ]: plt.figure(figsize=(10,10))
        afweather.unstack().plot(kind='bar');plt.title('Accident_severit, Weather_condition')
        print("1= Fatal  2= Severe  3= Slight")
        print("1:Fine no high winds,2:Raining no high winds,3:Snowing no high winds,4:Fine + hig

In [ ]: plt.figure(figsize=(10,10))
        aflight.unstack().plot(kind='bar');plt.title('Accident_severit, Light_condition')
        print("Accident Severity: 1= Fatal  2= Severe  3= Slight")
        print("""\nLight_condition:
        1        Daylight
        4        Darkness - lights lit
        5        Darkness - lights unlit
        6        Darkness - no lighting
        7        Darkness - lighting unknown
        """)

In [ ]: plt.figure(figsize=(10,10))
        afroadtype.unstack().plot(kind='bar');plt.title('Accident_severit, Road_Type')
        print("1= Fatal  2= Severe  3= Slight")
        print("""\nRoad_type
        1 Roundabout
        2 One way street
        3 Dual carriageway
        6 Single carriageway
        7 Slip road
        9 Unknown

        """)

In [ ]: plt.figure(figsize=(10,10))
        afurbanrural.unstack().plot(kind='bar',stacked = True)
        print("Accident severity: 1= Fatal  2= Severe  3= Slight")
        print("""\nUrban_Rural:
```

```
        1        Urban
        2        Rural
        """)
```

In [ ]:
```
plt.figure(figsize=(10,10))
afroadsurface.unstack().plot(kind='bar')
print("accident severity")
print("""Road_surface_conditions
1 Dry
2 Wet or damp
3 Snow
4 Frost or ice
5 Mud
""")
```

In [ ]:

In [ ]: