In [1]:
```
#!pip install geopy
```

In [2]:
```
#!pip install geocoder
```

In [3]:
```
#!pip install uszipcode
```

In [4]:
```
#!pip install arcgis
```

In [5]:
```
#!pip install --force-reinstall numpy==1.23.3
```

In [6]:
```
#import numpy
#numpy.version.version
```

In [7]:
```
#!pip install numba
```

In [856]:
```
#!pip install gmaps

```

...

In [790]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_validate, Shuffl

from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix, roc_au
plot_confusion_matrix, precision_recall_curve

from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, VotingClassifier, AdaBo
AdaBoostClassifier, GradientBoostingClassifier

from sklearn.cluster import KMeans
import xgboost
#import geopy
#from geopy.geocoders import Nominatim
#from arcgis.geocoding import reverse_geocode

```

Making GPU as processor

In [9]:

```
import numba
numba.__version__
```

Out[9]:

```
'0.56.2'
```

In [10]:

```
from numba import jit, cuda
import numpy as np
# to measure exec time
from timeit import default_timer as timer

# normal function to run on cpu
def func(a):
    for i in range(10000000):
        a[i]+= 1

# function optimized to run on gpu
#@jit(target_backend='cuda')
def func2(a):
    for i in range(10000000):
        a[i]+= 1
if __name__=="__main__":
    n = 10000000
    a = np.ones(n, dtype = np.float64)

    start = timer()
    func(a)
    print("without GPU:", timer()-start)

    start = timer()
    func2(a)
    print("with GPU:", timer()-start)
```

```
without GPU: 2.4438747999999997
with GPU: 2.406006399999999
```

Importing & Exploring Data

In [11]:

```
df_crashes = pd.read_csv('data/Traffic_Crashes_-_Crashes.csv')
df_crashes.head()
```

Out[11]:

	CRASH_RECORD_ID	RD_NO	CRASH_DATE_EST_I	CRASH_DA
0	062f5a6f6b87b762165d4da04d6d3a181385776a10b051...	JF378246	NaN	08/31/20 10:13:00 F
1	0115ade9a755e835255508463f7e9c4a9a0b47e9304238...	JF318029	NaN	07/15/20 12:45:00 A
2	017040c61958d2fa977c956b2bd2d6759ef7754496dc96...	JF324552	NaN	07/15/20 06:50:00 F
3	01aaa759c6bbefd0f584226fbd88bdc549de3ed1e46255...	JF319819	NaN	07/15/20 05:10:00 F
4	04f21d51f8189e34abf37c7973607fa076965d216b514f...	JC366684	NaN	07/22/20 12:00:00 F

5 rows × 49 columns

In [12]:

```
df_crashes.info()
```

...

In [13]:

```
df_crashes.TRAFFIC_CONTROL_DEVICE.value_counts()
```

Out[13]:

```
NO CONTROLS                 376542
TRAFFIC SIGNAL              181633
STOP SIGN/FLASHER            64989
UNKNOWN                      22416
OTHER                         4205
LANE USE MARKING              1226
YIELD                          913
OTHER REG. SIGN                683
OTHER WARNING SIGN             565
RAILROAD CROSSING GATE         423
PEDESTRIAN CROSSING SIGN       357
DELINEATORS                    247
POLICE/FLAGMAN                 238
SCHOOL ZONE                    236
FLASHING CONTROL SIGNAL        223
OTHER RAILROAD CROSSING        153
RR CROSSING SIGN                78
NO PASSING                      36
BICYCLE CROSSING SIGN           19
Name: TRAFFIC_CONTROL_DEVICE, dtype: int64
```

In [14]:

```
df_crashes.isna().sum()
```

Out[14]:

```
CRASH_RECORD_ID                       0
RD_NO                              4587
CRASH_DATE_EST_I                 605487
CRASH_DATE                            0
POSTED_SPEED_LIMIT                    0
TRAFFIC_CONTROL_DEVICE                0
DEVICE_CONDITION                      0
WEATHER_CONDITION                     0
LIGHTING_CONDITION                    0
FIRST_CRASH_TYPE                      0
TRAFFICWAY_TYPE                       0
LANE_CNT                         456191
ALIGNMENT                             0
ROADWAY_SURFACE_COND                  0
ROAD_DEFECT                           0
REPORT_TYPE                       17491
CRASH_TYPE                            0
INTERSECTION_RELATED_I           505314
NOT_RIGHT_OF_WAY_I               624336
HIT_AND_RUN_I                    453020
DAMAGE                                0
DATE_POLICE_NOTIFIED                  0
PRIM_CONTRIBUTORY_CAUSE               0
SEC_CONTRIBUTORY_CAUSE                0
STREET_NO                             0
STREET_DIRECTION                      4
STREET_NAME                           1
BEAT_OF_OCCURRENCE                    5
PHOTOS_TAKEN_I                   647098
STATEMENTS_TAKEN_I               641669
DOORING_I                        653127
WORK_ZONE_I                      651287
WORK_ZONE_TYPE                   652126
WORKERS_PRESENT_I                654185
NUM_UNITS                             0
MOST_SEVERE_INJURY                 1401
INJURIES_TOTAL                     1390
INJURIES_FATAL                     1390
INJURIES_INCAPACITATING            1390
INJURIES_NON_INCAPACITATING        1390
INJURIES_REPORTED_NOT_EVIDENT      1390
INJURIES_NO_INDICATION             1390
INJURIES_UNKNOWN                   1390
CRASH_HOUR                            0
CRASH_DAY_OF_WEEK                     0
CRASH_MONTH                           0
LATITUDE                           4074
LONGITUDE                          4074
LOCATION                           4074
dtype: int64
```

In [15]:

```python
# Keeping relevant Features
df_crashes_drop = df_crashes [[
    'CRASH_RECORD_ID',
#     'RD_NO',
    'CRASH_DATE',
    'POSTED_SPEED_LIMIT',
    'WEATHER_CONDITION',
#     'LIGHTING_CONDITION',
#     'FIRST_CRASH_TYPE',
    'ROADWAY_SURFACE_COND',
    'ROAD_DEFECT',
#     'CRASH_TYPE',
#     'DAMAGE',
#     'PRIM_CONTRIBUTORY_CAUSE',
#     'STREET_NAME',
#     'NUM_UNITS',
    'INJURIES_TOTAL',
    'INJURIES_FATAL',
    'CRASH_HOUR',
    'CRASH_DAY_OF_WEEK',
    'CRASH_MONTH',
    'LATITUDE',
    'LONGITUDE',
#     'LOCATION',
]]
print(df_crashes_drop.shape)
```

```
(655182, 13)
```

In [16]:

```python
df_crashes_drop.describe()
```

Out[16]:

	POSTED_SPEED_LIMIT	INJURIES_TOTAL	INJURIES_FATAL	CRASH_HOUR	CRASH_DAY_OF_
count	655182.000000	653792.000000	653792.000000	655182.000000	655182.
mean	28.356959	0.184946	0.001169	13.222685	4.
std	6.296888	0.558173	0.037166	5.549903	1.
min	0.000000	0.000000	0.000000	0.000000	1.
25%	30.000000	0.000000	0.000000	9.000000	2.
50%	30.000000	0.000000	0.000000	14.000000	4.
75%	30.000000	0.000000	0.000000	17.000000	6.
max	99.000000	21.000000	4.000000	23.000000	7.

In [17]:

```python
# Dropping Rows with Latitude & Longitude = 0
df_crashes_drop = df_crashes_drop[df_crashes_drop['LATITUDE'] != 0]
```

In [18]:

```python
df_crashes_drop['has_injuries'] = df_crashes_drop.INJURIES_TOTAL.apply(lambd
df_crashes_drop['has_fatality'] = df_crashes_drop.INJURIES_FATAL.apply(lambd
```

In [1025]:

```
# Visualizing the distribution of accidents by having injuries ( Alternative
crash_df_ = df_crashes_drop.groupby(by=['LONGITUDE','LATITUDE']).agg(crashes
crash_df_
```

Out[1025]:

	LONGITUDE	LATITUDE	crashes	has_injuries
0	-87.936193	41.960822	1	0
1	-87.935877	41.960761	1	0
2	-87.934763	41.960230	3	0
3	-87.934510	42.008051	1	0
4	-87.934014	41.959123	1	0
...	...	...	...	...
260229	-87.524674	41.702590	8	1
260230	-87.524646	41.698928	1	0
260231	-87.524640	41.703323	1	0
260232	-87.524589	41.702571	4	1
260233	-87.524587	41.703272	7	1

260234 rows × 4 columns

In [862]:

```
#!pip install plotly==5.8.0
```

```
Collecting plotly==5.8.0
  Downloading plotly-5.8.0-py2.py3-none-any.whl (15.2 MB)
Collecting tenacity>=6.2.0
  Downloading tenacity-8.1.0-py3-none-any.whl (23 kB)
Installing collected packages: tenacity, plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 4.11.0
    Uninstalling plotly-4.11.0:
      Successfully uninstalled plotly-4.11.0
Successfully installed plotly-5.8.0 tenacity-8.1.0
```

In [866]:

```
import folium
from folium.plugins import HeatMap
```

In [1028]:

```
lats_longs_weight = list(map(list, zip(crash_df_["LATITUDE"],
                          crash_df_["LONGITUDE"],
                          crash_df_["crashes"]
                         )
                    )
                )
```

In [1029]:

```python
map_obj = folium.Map(location = [41.874389144012255, -87.668751362594],tiles

HeatMap(lats_longs_weight, min_opacity=0.5,max_zoom = 40,radius=9,control=Tr
#folium.LayerControl().add_to(map_obj)
map_obj
```

Out[1029]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [21]:

```python
# Dropping unknown values
df_crashes_drop.replace({'UNKNOWN':np.nan}, inplace=True)
```

In [22]:

```python
# Dropping rows with lats and long =0
df_crashes_ = df_crashes_drop.dropna(subset = 'LATITUDE', axis = 0);
```

In [23]:

```python
df_vehicles = pd.read_csv('data/Traffic_Crashes_-_Vehicles.csv')
df_vehicles.head()
```

```
<ipython-input-23-fce9fee1d033>:1: DtypeWarning: Columns (19,21,40,41,42,44,48,
49,50,53,55,58,59,61,71) have mixed types. Specify dtype option on import or se
t low_memory=False.
  df_vehicles = pd.read_csv('data/Traffic_Crashes_-_Vehicles.csv')
```

Out[23]:

	CRASH_UNIT_ID	CRASH_RECORD_ID	RD_NO	CRASH_DATE
0	829999	24ddf9fd8542199d832e1c223cc474e5601b356f1d77a6...	JD124535	01/22/2020 06:25:00 AM
1	749947	81dc0de2ed92aa62baccab641fa377be7feb1cc47e6554...	JC451435	09/28/2019 03:30:00 AM
2	749949	81dc0de2ed92aa62baccab641fa377be7feb1cc47e6554...	JC451435	09/28/2019 03:30:00 AM
3	749950	81dc0de2ed92aa62baccab641fa377be7feb1cc47e6554...	JC451435	09/28/2019 03:30:00 AM
4	871921	af84fb5c8d996fcd3aefd36593c3a02e6e7509eeb27568...	JD208731	04/13/2020 10:50:00 PM

5 rows × 72 columns

In [24]:

```
df_vehicles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1335230 entries, 0 to 1335229
Data columns (total 72 columns):
 #   Column                     Non-Null Count    Dtype  
---  ------                     --------------    -----  
 0   CRASH_UNIT_ID              1335230 non-null  int64  
 1   CRASH_RECORD_ID            1335230 non-null  object 
 2   RD_NO                      1325924 non-null  object 
 3   CRASH_DATE                 1335230 non-null  object 
 4   UNIT_NO                    1335230 non-null  int64  
 5   UNIT_TYPE                  1333361 non-null  object 
 6   NUM_PASSENGERS             198353 non-null   float64
 7   VEHICLE_ID                 1304967 non-null  float64
 8   CMRC_VEH_I                 24927 non-null    object 
 9   MAKE                       1304962 non-null  object 
 10  MODEL                      1304818 non-null  object 
 11  LIC_PLATE_STATE            1187047 non-null  object 
 12  VEHICLE_YEAR               1092432 non-null  float64
 13  VEHICLE_DEFECT             1304967 non-null  object 
 14  VEHICLE_TYPE               1304967 non-null  object 
 15  VEHICLE_USE                1304967 non-null  object 
 16  TRAVEL_DIRECTION           1304967 non-null  object 
 17  MANEUVER                   1304967 non-null  object 
 18  TOWED_I                    162050 non-null   object 
 19  FIRE_I                     1062 non-null     object 
 20  OCCUPANT_CNT               1304967 non-null  float64
 21  EXCEED_SPEED_LIMIT_I       2397 non-null     object 
 22  TOWED_BY                   120536 non-null   object 
 23  TOWED_TO                   74156 non-null    object 
 24  AREA_00_I                  47900 non-null    object 
 25  AREA_01_I                  351739 non-null   object 
 26  AREA_02_I                  216469 non-null   object 
 27  AREA_03_I                  126691 non-null   object 
 28  AREA_04_I                  128805 non-null   object 
 29  AREA_05_I                  199196 non-null   object 
 30  AREA_06_I                  206230 non-null   object 
 31  AREA_07_I                  185253 non-null   object 
 32  AREA_08_I                  202505 non-null   object 
 33  AREA_09_I                  76788 non-null    object 
 34  AREA_10_I                  110769 non-null   object 
 35  AREA_11_I                  217168 non-null   object 
 36  AREA_12_I                  213818 non-null   object 
 37  AREA_99_I                  146804 non-null   object 
 38  FIRST_CONTACT_POINT        1293476 non-null  object 
 39  CMV_ID                     14038 non-null    float64
 40  USDOT_NO                   8044 non-null     object 
 41  CCMC_NO                    1755 non-null     object 
 42  ILCC_NO                    1230 non-null     object 
 43  COMMERCIAL_SRC             9556 non-null     object 
 44  GVWR                       7988 non-null     object 
 45  CARRIER_NAME               13428 non-null    object 
 46  CARRIER_STATE              12645 non-null    object 
 47  CARRIER_CITY               12425 non-null    object 
 48  HAZMAT_PLACARDS_I          276 non-null      object 
 49  HAZMAT_NAME                51 non-null       object 
 50  UN_NO                     489 non-null      object 
 51  HAZMAT_PRESENT_I          10282 non-null    object 
 52  HAZMAT_REPORT_I           9972 non-null     object 
 53  HAZMAT_REPORT_NO          1 non-null        object 
 54  MCS_REPORT_I              10033 non-null    object 
 55  MCS_REPORT_NO             6 non-null        object 
 56  HAZMAT_VIO_CAUSE_CRASH_I  10130 non-null    object 
 57  MCS_VIO_CAUSE_CRASH_I     9946 non-null     object 
 58  IDOT_PERMIT_NO            793 non-null      object 
 59  WIDE_LOAD_I               123 non-null      object 
 60  TRAILER1_WIDTH            2584 non-null     object 
 61  TRAILER2_WIDTH            303 non-null      object 
 62  TRAILER1_LENGTH           2124 non-null     float64
 63  TRAILER2_LENGTH           60 non-null       float64
 64  TOTAL_VEHICLE_LENGTH      2577 non-null     float64
 65  AXLE_CNT                  3786 non-null     float64
 66  VEHICLE_CONFIG            11649 non-null    object 
 67  CARGO_BODY_TYPE           11148 non-null    object 
 68  LOAD_TYPE                 10655 non-null    object 
 69  HAZMAT_OUT_OF_SERVICE_I   9674 non-null     object 
 70  MCS_OUT_OF_SERVICE_I      9921 non-null     object 
 71  HAZMAT_CLASS              939 non-null      object 
dtypes: float64(9), int64(2), object(61)
memory usage: 733.5+ MB
```

In [25]:

```python
# Keeping Relevant Features
df_vehicles_drop = df_vehicles [[
    'CRASH_RECORD_ID',
#     'RD_NO',
#     'CRASH_DATE',
    'VEHICLE_ID',
#     'MAKE',
#     'MODEL',
#     'LIC_PLATE_STATE',
    'VEHICLE_YEAR',
#     'VEHICLE_DEFECT',
#     'VEHICLE_TYPE',
#     'VEHICLE_USE',
#     'TRAVEL_DIRECTION',
    'OCCUPANT_CNT',
#     'VEHICLE_CONFIG',
]]
print(df_vehicles_drop.shape)
```

```
(1335230, 4)
```

In [26]:
```
df_vehicles.VEHICLE_CONFIG.value_counts()
```

Out[26]:
```
TRACTOR/SEMI-TRAILER                     4572
SINGLE UNIT TRUCK, 2 AXLES, 6 TIRES      2337
BUS                                      1805
TRUCK/TRACTOR                             944
TRUCK/TRAILER                             770
UNKNOWN HEAVY TRUCK                       673
SINGLE UNIT TRUCK, 3 OR MORE AXLES        516
TRACTOR/DOUBLES                            32
Name: VEHICLE_CONFIG, dtype: int64
```

In [27]:
```
df_vehicles_drop.describe()
```

Out[27]:

	VEHICLE_ID	VEHICLE_YEAR	OCCUPANT_CNT
count	1.304967e+06	1.092432e+06	1.304967e+06
mean	6.789888e+05	2.013626e+03	1.079589e+00
std	3.917114e+05	1.439261e+02	7.843770e-01
min	2.000000e+00	1.900000e+03	0.000000e+00
25%	3.400725e+05	2.006000e+03	1.000000e+00
50%	6.794450e+05	2.012000e+03	1.000000e+00
75%	1.017828e+06	2.016000e+03	1.000000e+00
max	1.358762e+06	9.999000e+03	9.900000e+01

In [28]:
```
# Dropping rows with vehicle year bigger than 2022 and below 1970
df_vehicles_drop1 = df_vehicles_drop[(df_vehicles_drop['VEHICLE_YEAR'] >= 19
```

In [31]:
```
# Dropping Vehicles with 0 Occupant (Parked Cars)
df_vehicles_w_occ = df_vehicles_drop1[df_vehicles_drop.OCCUPANT_CNT != 0.0]
print(df_vehicles_w_occ.shape)
```

```
(1160416, 4)

<ipython-input-31-a0910bced00f>:2: UserWarning: Boolean Series key will be rein
dexed to match DataFrame index.
  df_vehicles_w_occ = df_vehicles_drop1[df_vehicles_drop.OCCUPANT_CNT != 0.0]
```

In [32]:
```
df_vehicles_w_occ.isna().sum()
```

Out[32]:
```
CRASH_RECORD_ID         0
VEHICLE_ID          30263
VEHICLE_YEAR       231819
OCCUPANT_CNT        30263
dtype: int64
```

In [33]:

```python
# Replacing all UNKNOWN values to missing values
df_vehicles_w_occ.replace({'UNKNOWN':np.nan}, inplace=True)
```

```
<ipython-input-33-fdbee41cc7b6>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_vehicles_w_occ.replace({'UNKNOWN':np.nan}, inplace=True)
```

In [34]:

```python
df_people = pd.read_csv('data/Traffic_Crashes_-_PEOPLE.csv')
df_people.head()

```

```
<ipython-input-34-78bd59adc723>:1: DtypeWarning: Columns (20,24,25,26,29) have mixed types. Specify dtype option on import or set low_memory=False.
  df_people = pd.read_csv('data/Traffic_Crashes_-_PEOPLE.csv')
```

Out[34]:

	PERSON_ID	PERSON_TYPE	CRASH_RECORD_ID	RD_NO	VEH
0	O749947	DRIVER	81dc0de2ed92aa62baccab641fa377be7feb1cc47e6554...	JC451435	8
1	O871921	DRIVER	af84fb5c8d996fcd3aefd36593c3a02e6e7509eeb27568...	JD208731	8
2	O10018	DRIVER	71162af7bf22799b776547132ebf134b5b438dcf3dac6b...	HY484534	
3	O10038	DRIVER	c21c476e2ccc41af550b5d858d22aaac4ffc88745a1700...	HY484750	
4	O10039	DRIVER	eb390a4c8e114c69488f5fb8a097fe629f5a92fd528cf4...	HY484778	

5 rows × 30 columns

In [35]:

```
pd.set_option('display.max_rows', 100)
df_people.isna().sum()
```

Out[35]:

```
PERSON_ID                        0
PERSON_TYPE                      0
CRASH_RECORD_ID                  0
RD_NO                        10002
VEHICLE_ID                   28251
CRASH_DATE                       0
SEAT_NO                    1145710
CITY                        386612
STATE                       372988
ZIPCODE                     477431
SEX                          22243
AGE                         417101
DRIVERS_LICENSE_STATE       592164
DRIVERS_LICENSE_CLASS       719487
SAFETY_EQUIPMENT              4087
AIRBAG_DEPLOYED              27110
EJECTION                     17338
INJURY_CLASSIFICATION          632
HOSPITAL                   1186950
EMS_AGENCY                 1281208
EMS_RUN_NO                 1411946
DRIVER_ACTION               294521
DRIVER_VISION               294925
PHYSICAL_CONDITION          293716
PEDPEDAL_ACTION            1410885
PEDPEDAL_VISIBILITY        1410940
PEDPEDAL_LOCATION          1410889
BAC_RESULT                  293506
BAC_RESULT VALUE           1435858
CELL_PHONE_USE             1436454
dtype: int64
```

In [36]:

```
# Keeping Relevant Features
df_people_drop = df_people [[
    'CRASH_RECORD_ID',
#     'RD_NO',
#     'PERSON_ID',
    'PERSON_TYPE',
    'VEHICLE_ID',
#     'CITY',
#     'STATE',
#     'ZIPCODE',
    'SEX',
    'AGE',
#     'DRIVERS_LICENSE_STATE',
#     'DRIVERS_LICENSE_CLASS',
    'SAFETY_EQUIPMENT',
    'AIRBAG_DEPLOYED',
#     'EJECTION',
#     'INJURY_CLASSIFICATION',
#     'DRIVER_VISION',
#     'DRIVER_ACTION',
#     'PHYSICAL_CONDITION',
#     'PEDPEDAL_ACTION',
#     'PEDPEDAL_VISIBILITY',
#     'PEDPEDAL_LOCATION',
#     'BAC_RESULT',
#     'BAC_RESULT VALUE',
#     'CELL_PHONE_USE',
]]
print(df_people_drop.shape)
```

```
(1437611, 7)
```

In [37]:

```
# Filtering the data with Drivers only
df_people_driver = df_people_drop[df_people_drop.PERSON_TYPE == 'DRIVER']
print(df_people_driver.shape)
```

```
(1117947, 7)
```

In [38]:

```
df_people_driver.isna().sum()
```

Out[38]:

```
CRASH_RECORD_ID          0
PERSON_TYPE              0
VEHICLE_ID             632
SEX                     97
AGE                 299295
SAFETY_EQUIPMENT         0
AIRBAG_DEPLOYED          0
dtype: int64
```

In [39]:
```
df_people_driver.describe()
```

Out[39]:

	VEHICLE_ID	AGE
count	1.117315e+06	818652.000000
mean	6.738542e+05	40.013475
std	3.924537e+05	15.836726
min	2.000000e+00	-177.000000
25%	3.338875e+05	27.000000
50%	6.705940e+05	37.000000
75%	1.014066e+06	51.000000
max	1.358762e+06	110.000000

In [40]:
```
# Dropping rows with driver age bigger than 90 and below 18
df_people_driver_age = df_people_driver[(df_people_driver['AGE'] >= 18) & (d
```

In [41]:
```
# Replacing all UNKNOWN values to missing values
#Unknown = ['UNKNOWN','USAGE UNKNOWN', 'DEPLOYMENT UNKNOWN']
df_vehicles_w_occ.replace({'Unknown' :np.nan , 'USAGE UNKNOWN' :np.nan , 'DE
```

```
<ipython-input-41-fad22d175994>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  df_vehicles_w_occ.replace({'Unknown' :np.nan , 'USAGE UNKNOWN' :np.nan , 'DEP
LOYMENT UNKNOWN':np.nan }, inplace=True)
```

Merging Tables

In [42]:
```
merged = df_crashes_drop.merge(df_vehicles_w_occ, on='CRASH_RECORD_ID')
print(merged.shape)
```

```
(1160337, 18)
```

In [43]:
```
df = merged.merge(df_people_driver_age, on=['VEHICLE_ID','CRASH_RECORD_ID'])
print(df.shape)
```

```
(1099817, 23)
```

In [44]:
```python
df.head()
```
Out[44]:

	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT	WEA
0	062f5a6f6b87b762165d4da04d6d3a181385776a10b051...	08/31/2022 10:13:00 PM	30	
1	062f5a6f6b87b762165d4da04d6d3a181385776a10b051...	08/31/2022 10:13:00 PM	30	
2	0115ade9a755e835255508463f7e9c4a9a0b47e9304238...	07/15/2022 12:45:00 AM	30	
3	0115ade9a755e835255508463f7e9c4a9a0b47e9304238...	07/15/2022 12:45:00 AM	30	
4	017040c61958d2fa977c956b2bd2d6759ef7754496dc96...	07/15/2022 06:50:00 PM	30	

5 rows × 23 columns

In [45]:
```python
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1099817 entries, 0 to 1099816
Data columns (total 23 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   CRASH_RECORD_ID        1099817 non-null  object
 1   CRASH_DATE             1099817 non-null  object
 2   POSTED_SPEED_LIMIT     1099817 non-null  int64
 3   WEATHER_CONDITION      1055149 non-null  object
 4   ROADWAY_SURFACE_COND   1025920 non-null  object
 5   ROAD_DEFECT            926974 non-null   object
 6   INJURIES_TOTAL         1099817 non-null  float64
 7   INJURIES_FATAL         1099817 non-null  float64
 8   CRASH_HOUR             1099817 non-null  int64
 9   CRASH_DAY_OF_WEEK      1099817 non-null  int64
 10  CRASH_MONTH            1099817 non-null  int64
 11  LATITUDE               1092714 non-null  float64
 12  LONGITUDE              1092714 non-null  float64
 13  has_injuries           1099817 non-null  int64
 14  has_fatality           1099817 non-null  int64
```

In [46]:

```python
# creating bins for times
# 0-6 = Late Night/Early Morning
# 6-12 = Morning
# 12-18 = Afternoon/Rush Hour
# 18-23 = Late Evening
df['time_bins'] = pd.cut(x=df['CRASH_HOUR'], bins = [-1,6,12,18,24],
                         labels = ['Late Night/Early Morning',
                        'Morning', 'Afternoon/Rush Hour','Late Evening'])
df.head(50)
```

Out[46]:

	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT	W
0	062f5a6f6b87b762165d4da04d6d3a181385776a10b051...	08/31/2022 10:13:00 PM	30	
1	062f5a6f6b87b762165d4da04d6d3a181385776a10b051...	08/31/2022 10:13:00 PM	30	
2	0115ade9a755e835255508463f7e9c4a9a0b47e9304238...	07/15/2022 12:45:00 AM	30	
3	0115ade9a755e835255508463f7e9c4a9a0b47e9304238...	07/15/2022 12:45:00 AM	30	
4	017040c61958d2fa977c956b2bd2d6759ef7754496dc96...	07/15/2022 06:50:00 PM	30	
5	017040c61958d2fa977c956b2bd2d6759ef7754496dc96...	07/15/2022 06:50:00 PM	30	
6	01aaa759c6bbcfd0f584226fbd88bdc549de3ed1c46255...	07/15/2022	40	

In [47]:

```python
# Dropping Unnecassary Features
df_relv = df.drop([
    'CRASH_RECORD_ID',
#     'RD_NO',
#     'PERSON_ID',
    'CRASH_DATE',
    'VEHICLE_ID',
#     'CITY',
#     'STATE',
#     'ZIPCODE',
    'PERSON_TYPE',
    'OCCUPANT_CNT',
#     'has_injuries',
#     'has_fatality',
#     'LONGITUDE',
#     'LATITUDE',
    'ROAD_DEFECT',
#     'LIC_PLATE_STATE',
#     'TRAVEL_DIRECTION',
#     'DRIVERS_LICENSE_STATE',
#     'INJURY_CLASSIFICATION',
#     'DRIVER_ACTION',
#     'PHYSICAL_CONDITION'],
],axis=1)
```

In [48]:

```
df_drop_missing = df_relv.dropna(subset = ['LATITUDE','LONGITUDE'], axis = 0
df_drop_missing.shape
```

Out[48]:

```
(1092714, 18)
```

In [49]:

```
df_drop_missing.head()
```

Out[49]:

	POSTED_SPEED_LIMIT	WEATHER_CONDITION	ROADWAY_SURFACE_COND	INJURIES_TOTAL	IN
0	30	CLEAR	DRY	2.0	
1	30	CLEAR	DRY	2.0	
2	30	CLEAR	DRY	0.0	
3	30	CLEAR	DRY	0.0	
4	30	CLEAR	DRY	0.0	

In [50]:

```
df1 = df_drop_missing[['LATITUDE',"LONGITUDE"]]
df1
```

Out[50]:

	LATITUDE	LONGITUDE
0	41.959389	-87.747348
1	41.959389	-87.747348
2	41.886336	-87.716203
3	41.886336	-87.716203
4	41.925111	-87.667997
...	...	...
1092709	41.735671	-87.663670
1092710	41.735671	-87.663670
1092711	41.877164	-87.720464
1092712	41.877164	-87.720464
1092713	41.989257	-87.776270

1092714 rows × 2 columns

In [51]:

```python
# Biinning Locations
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 50, init ='k-means++')
kmeans.fit(df1[df1.columns]) # Compute k-means clustering.
df1['cluster_label'] = kmeans.fit_predict(df1[df1.columns])
#centers = kmeans.cluster_centers_ # Coordinates of cluster centers.
#labels = kmeans.predict(df1[df1.columns]) # Labels of each point
df1.head(20)
```

```
<ipython-input-51-f4dd10255c2f>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df1['cluster_label'] = kmeans.fit_predict(df1[df1.columns])
```

Out[51]:

	LATITUDE	LONGITUDE	cluster_label
0	41.959389	-87.747348	15
1	41.959389	-87.747348	15
2	41.886336	-87.716203	39
3	41.886336	-87.716203	39
4	41.925111	-87.667997	43
5	41.925111	-87.667997	43
6	41.975826	-87.650420	9
7	41.975826	-87.650420	9
8	41.737337	-87.563560	19
9	41.737337	-87.563560	19
10	41.944199	-87.747157	31
11	41.944199	-87.747157	31
12	41.807712	-87.744440	7
13	41.855974	-87.663860	34
14	41.730216	-87.548387	41
15	41.956477	-87.785397	8
16	41.956477	-87.785397	8
17	41.807856	-87.733435	7
18	41.807856	-87.733435	7
19	41.877626	-87.629862	20

In [54]:
```
df_drop_missing['loc_clusters'] = df1['cluster_label']
```

In [55]:
```
df_drop_missing.isna().sum()
```

Out[55]:
```
POSTED_SPEED_LIMIT           0
WEATHER_CONDITION        44507
ROADWAY_SURFACE_COND     73591
INJURIES_TOTAL               0
INJURIES_FATAL               0
CRASH_HOUR                   0
CRASH_DAY_OF_WEEK            0
CRASH_MONTH                  0
LATITUDE                     0
LONGITUDE                    0
has_injuries                 0
has_fatality                 0
VEHICLE_YEAR            199814
SEX                         97
AGE                     297434
SAFETY_EQUIPMENT             0
AIRBAG_DEPLOYED              0
time_bins                    0
loc_clusters                 0
dtype: int64
```

In [56]:
```
# drop missing rows
df_drop_missing2 = df_drop_missing.dropna(subset= ['AGE','VEHICLE_YEAR','ROA
print(df_drop_missing2.isna().sum())
print(df_drop_missing2.shape)
```

```
POSTED_SPEED_LIMIT      0
WEATHER_CONDITION       0
ROADWAY_SURFACE_COND    0
INJURIES_TOTAL          0
INJURIES_FATAL          0
CRASH_HOUR              0
CRASH_DAY_OF_WEEK       0
CRASH_MONTH             0
LATITUDE                0
LONGITUDE               0
has_injuries            0
has_fatality            0
VEHICLE_YEAR            0
SEX                     0
AGE                     0
SAFETY_EQUIPMENT        0
AIRBAG_DEPLOYED         0
time_bins               0
loc_clusters            0
dtype: int64
(715945, 19)
```

In [57]:

```
# Dropping outliers
counts = df_drop_missing2['SAFETY_EQUIPMENT'].value_counts()
df_drop_missing2 = df_drop_missing2[~df_drop_missing2['SAFETY_EQUIPMENT'].is
```

In [58]:

```
df_drop_missing2.SAFETY_EQUIPMENT.value_counts()
```

Out[58]:

```
SAFETY BELT USED                        467974
USAGE UNKNOWN                           225687
NONE PRESENT                             16523
SAFETY BELT NOT USED                      3573
HELMET NOT USED                            931
DOT COMPLIANT MOTORCYCLE HELMET            679
HELMET USED                                391
NOT DOT COMPLIANT MOTORCYCLE HELMET        104
SHOULD/LAP BELT USED IMPROPERLY             79
Name: SAFETY_EQUIPMENT, dtype: int64
```

In [59]:

```
# Dropping outliers

counts = df_drop_missing2['WEATHER_CONDITION'].value_counts()
df_drop_missing2 = df_drop_missing2[~df_drop_missing2['WEATHER_CONDITION'].i
```

In [60]:

```
df_drop_missing2.to_csv('data/merged.csv')
```

In [419]:

```
# Sampling the population to make the data smaller
weights = {0 : 0.15, 1 : 0.85 }
df_drop_missing2['weights'] = df_drop_missing2['has_injuries'].apply(lambda
df_drop_missing2.head()
```

Out[419]:

	POSTED_SPEED_LIMIT	WEATHER_CONDITION	ROADWAY_SURFACE_COND	INJURIES_TOTAL	IN
0	30	CLEAR	DRY	2.0	
1	30	CLEAR	DRY	0.0	
2	30	CLEAR	DRY	0.0	
3	30	CLEAR	DRY	0.0	
4	40	CLOUDY/OVERCAST	DRY	0.0	

In [420]:

```
# Sampling the population to make the data smaller

df_bal = df_drop_missing2.sample(frac = 0.1, weights='weights',random_state
df_bal.has_injuries.value_counts()

```

Out[420]:

```
0    36858
1    34736
Name: has_injuries, dtype: int64
```

In [421]:

```
for col in df_reduced.columns:
    try:
        print(col, df_reduced[col].value_counts(dropna=False)[:20])
    except:
        print(col, df_reduced[col].value_counts())
        # If there aren't 5+ unique values for a column the first print stat
        # will throw an error for an invalid idx slice
    print('\n') # Break up the output between columns
```

```
POSTED_SPEED_LIMIT 30    54854
35     5462
25     3856
20     2079
15     1806
10     1060
40      869
0       631
45      536
5       316
55       46
3        23
50       19
39        8
9         6
65        3
2         3
34        3
60        3
```

Binning Imbalanced Features

Posted Speed Limit

In [422]:

```
# creating bins and label, previewing data
df_bal['speed_limit'] = pd.cut(x=df_bal['POSTED_SPEED_LIMIT'], bins = [-1,15
                          labels = ['0-15', '16-25',
                         '26-40', '41+'])
df_bal.speed_limit.value_counts()
```

Out[422]:

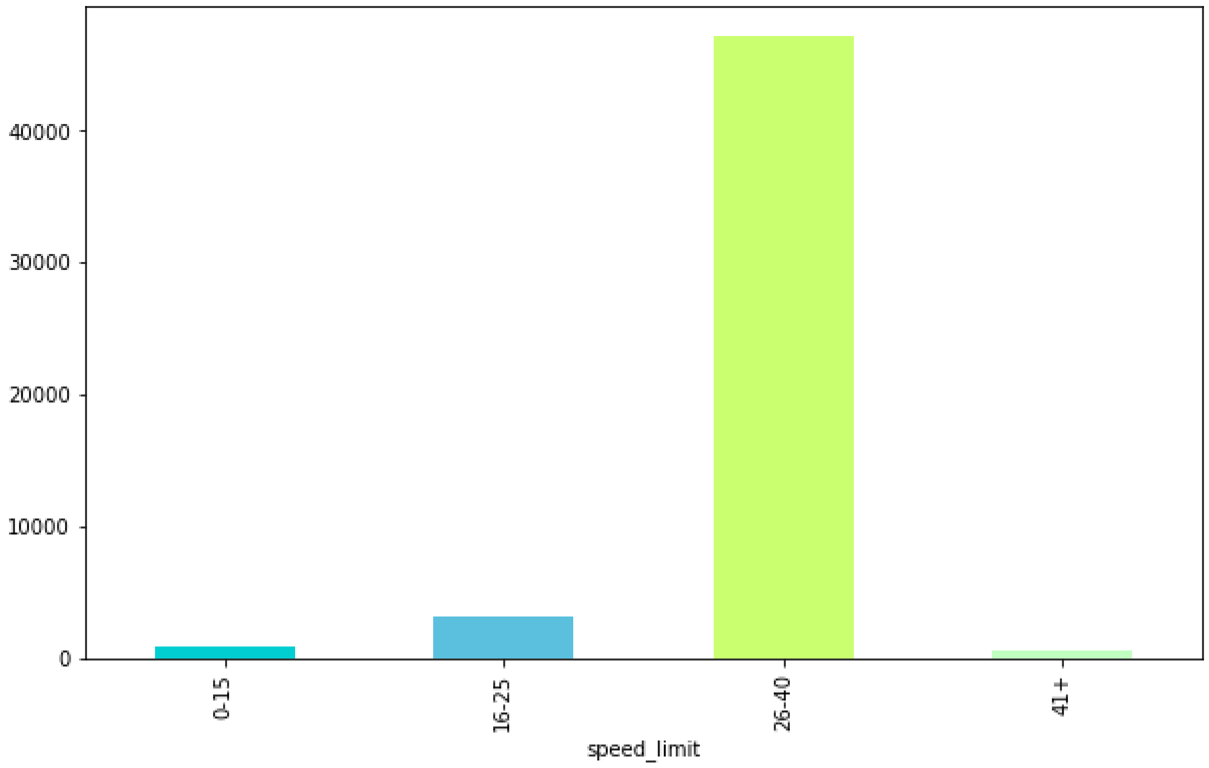
```
26-40    62641
16-25     5474
0-15      2820
41+        659
Name: speed_limit, dtype: int64
```

In [523]:

```
df_bal.groupby(['speed_limit']).INJURIES_TOTAL.sum().plot(kind='bar', color=
```

Out[523]:

```
<AxesSubplot:xlabel='speed_limit'>
```



Vehicle Year

In [513]:

```
# creating bins and labels, preview data
df_bal['vehicle_year'] = pd.cut(x=df_bal['VEHICLE_YEAR'], bins = [0,2000,200
                          labels = ['2000 & under', '2001-2005','2006-2012','

df_bal.vehicle_year.value_counts()
```

Out[513]:

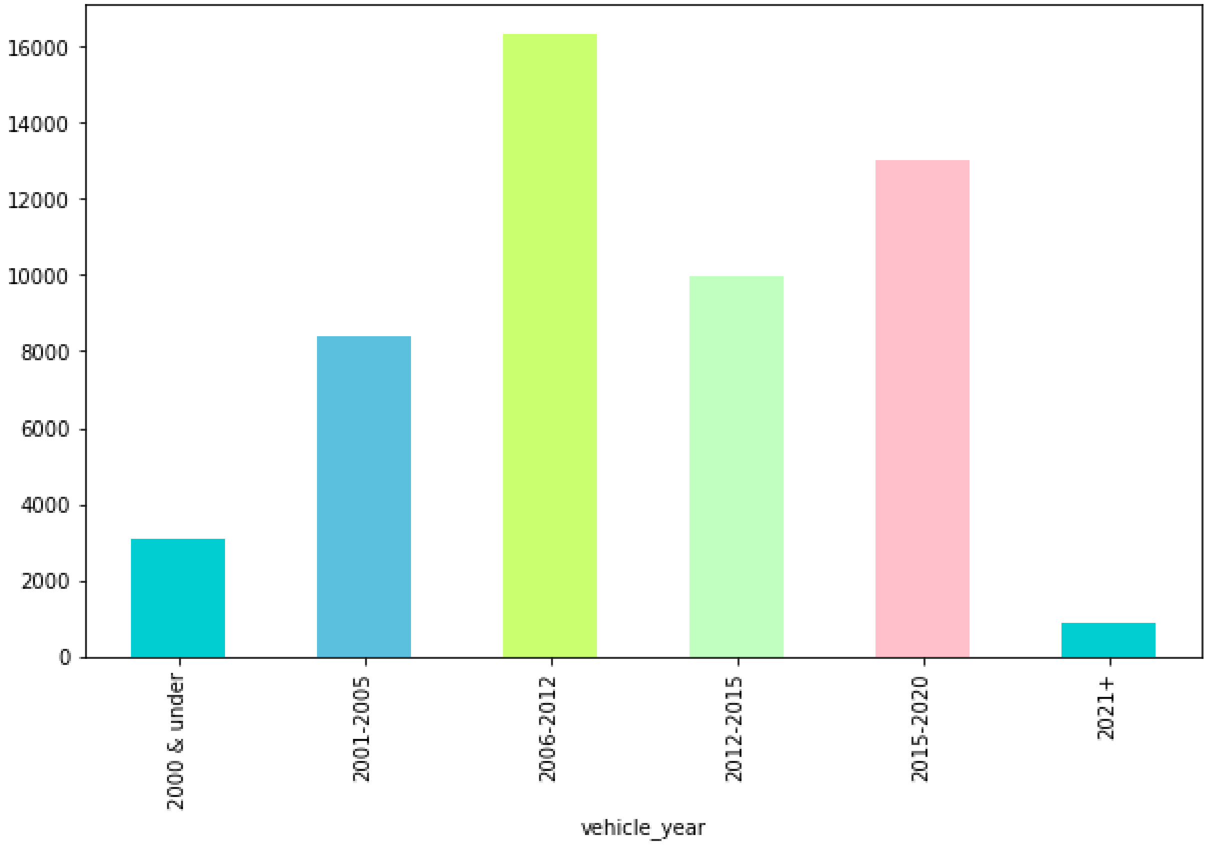
```
2006-2012       22414
2015-2020       18033
2012-2015       14768
2001-2005       10815
2000 & under     4143
2021+            1245
Name: vehicle_year, dtype: int64
```

In [514]:

```python
df_bal.groupby(['vehicle_year']).INJURIES_TOTAL.sum().plot(kind='bar', color
```

Out[514]:

```
<AxesSubplot:xlabel='vehicle_year'>
```



AGE

In [499]:

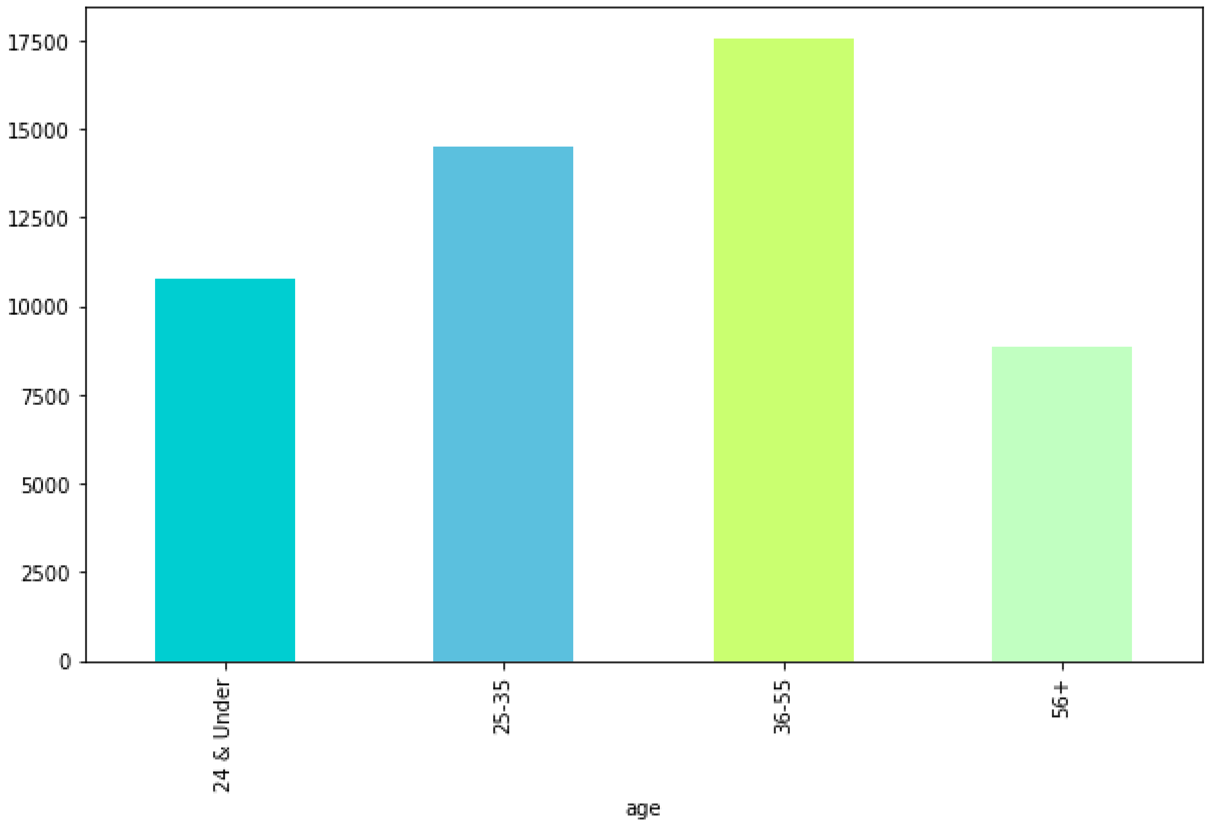
```python
# creating bins and labels, preview data
df_bal['age'] = pd.cut(x=df_bal['AGE'], bins = [0,25,35,55,100],
                             labels = ['24 & Under',
                            '25-35', '36-55','56+'])
df_bal.age.value_counts()

```

Out[499]:

```
36-55         25191
25-35         19381
56+           13426
24 & Under    13420
Name: age, dtype: int64
```

```
In [500]: df_bal.groupby(['age']).INJURIES_TOTAL.sum().plot(kind='bar', color=['#00CED
```

```
Out[500]: <AxesSubplot:xlabel='age'>
```



Creating bins and labels, preview data

AIRBAG

In [425]:

```
# creating bins and labels, preview data
Airbag_map = {'DID NOT DEPLOY': 'DID NOT DEPLOY',
                'NOT APPLICABLE': 'NOT APPLICABLE/UNKNOWN',
                'DEPLOYMENT UNKNOWN': 'NOT APPLICABLE/UNKNOWN',
                'DEPLOYED, FRONT': 'DEPLOYED',
                'DEPLOYED, COMBINATION': 'DEPLOYED',
                'DEPLOYED, SIDE': 'DEPLOYED',
                'DEPLOYED OTHER (KNEE, AIR, BELT, ETC.)': 'DEPLOYED'}
df_bal.AIRBAG_DEPLOYED = df_bal.AIRBAG_DEPLOYED.map(Airbag_map)
print(df_bal.AIRBAG_DEPLOYED.value_counts())
#print(df_bal.Airbag.value_counts())
```

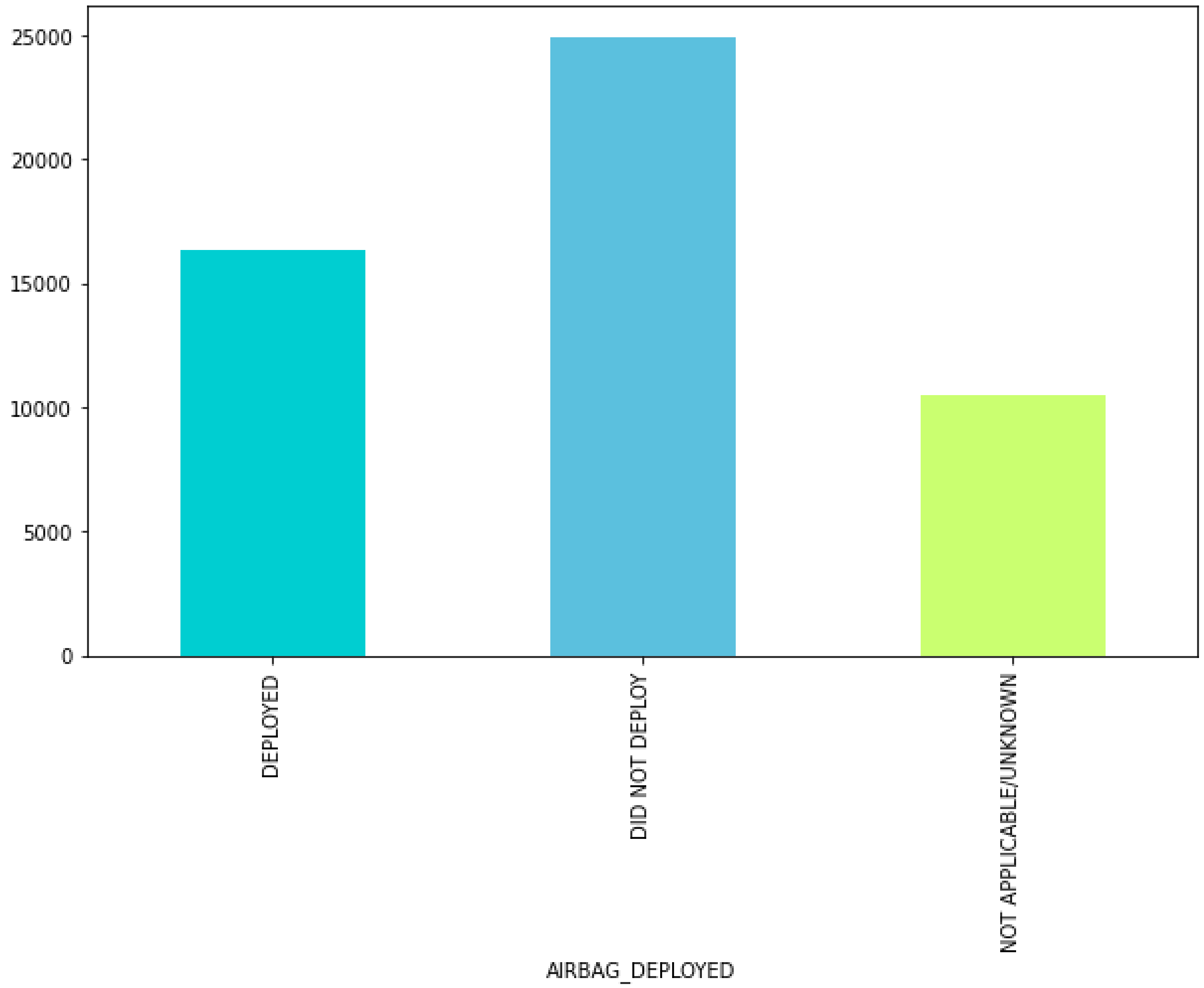
```
DID NOT DEPLOY            44373
NOT APPLICABLE/UNKNOWN    16305
DEPLOYED                  10916
Name: AIRBAG_DEPLOYED, dtype: int64
```

In [520]:

```
df_bal.groupby(['AIRBAG_DEPLOYED']).INJURIES_TOTAL.sum().plot(kind='bar', co
```

Out[520]:

```
<AxesSubplot:xlabel='AIRBAG_DEPLOYED'>
```



SAFETY EQ

In [426]:

```
safety_map = {'USAGE UNKNOWN': 'USAGE UNKNOWN',
                'SAFETY BELT USED': 'SAFETY EQUIPMENT USED',
                'NONE PRESENT': 'NONE PRESENT/USED',
                'HELMET NOT USED': 'NONE PRESENT/USED',
                'HELMET USED': 'SAFETY EQUIPMENT USED',
                'SAFETY BELT NOT USED': 'NONE PRESENT/USED',
                'NOT DOT COMPLIANT MOTORCYCLE HELMET ': 'NONE PRESENT/USED',
                'DOT COMPLIANT MOTORCYCLE HELMET ': 'SAFETY EQUIPMENT USED',
                'SHOULD/LAP BELT USED IMPROPERLY': 'NONE PRESENT/USED'}

df_bal.SAFETY_EQUIPMENT = df_bal.SAFETY_EQUIPMENT.map(safety_map)
df_bal.SAFETY_EQUIPMENT.value_counts()
```

Out[426]:

```
SAFETY EQUIPMENT USED    44960
USAGE UNKNOWN            24267
NONE PRESENT/USED         2191
Name: SAFETY_EQUIPMENT, dtype: int64
```

In [ ]:

```

```

Weather Condition

In [427]:

```
weather_map = {'CLEAR': 'CLEAR',
               'RAIN': 'RAIN/CLOUDY/OTHER',
               'CLOUDY/OVERCAST': 'RAIN/CLOUDY/OTHER',
               'SNOW': 'RAIN/CLOUDY/OTHER',
               'OTHER': 'RAIN/CLOUDY/OTHER',
               'SLEET/HAIL': 'RAIN/CLOUDY/OTHER',
               'FOG/SMOKE/HAZE': 'RAIN/CLOUDY/OTHER',
               'FREEZING RAIN/DRIZZLE': 'RAIN/CLOUDY/OTHER',
               'BLOWING SNOW': 'RAIN/CLOUDY/OTHER',
               'SEVERE CROSS WIND GATE': 'RAIN/CLOUDY/OTHER'}

df_bal.WEATHER_CONDITION = df_bal.WEATHER_CONDITION.map(weather_map)
df_bal.WEATHER_CONDITION.value_counts()
```

Out[427]:

```
CLEAR                58597
RAIN/CLOUDY/OTHER    12997
Name: WEATHER_CONDITION, dtype: int64
```

In [435]:

```
df_bal = df_bal.dropna( subset ='SAFETY_EQUIPMENT', axis=0)
```

In [436]:

```
df_bal.isna().sum()
```

Out[436]:

```
POSTED_SPEED_LIMIT      0
WEATHER_CONDITION       0
ROADWAY_SURFACE_COND    0
INJURIES_TOTAL          0
INJURIES_FATAL          0
CRASH_HOUR              0
CRASH_DAY_OF_WEEK       0
CRASH_MONTH             0
LATITUDE                0
LONGITUDE               0
has_injuries            0
has_fatality            0
VEHICLE_YEAR            0
SEX                     0
AGE                     0
SAFETY_EQUIPMENT        0
AIRBAG_DEPLOYED         0
time_bins               0
loc_clusters            0
weights                 0
speed_limit             0
vehicle_year            0
age                     0
dtype: int64
```

In [ ]:

```

```

Visualization

In [827]:

```
plt.figure(figsize=(10,10))
sns.countplot(x="CRASH_DAY_OF_WEEK", hue="has_injuries", data=df_bal)
```

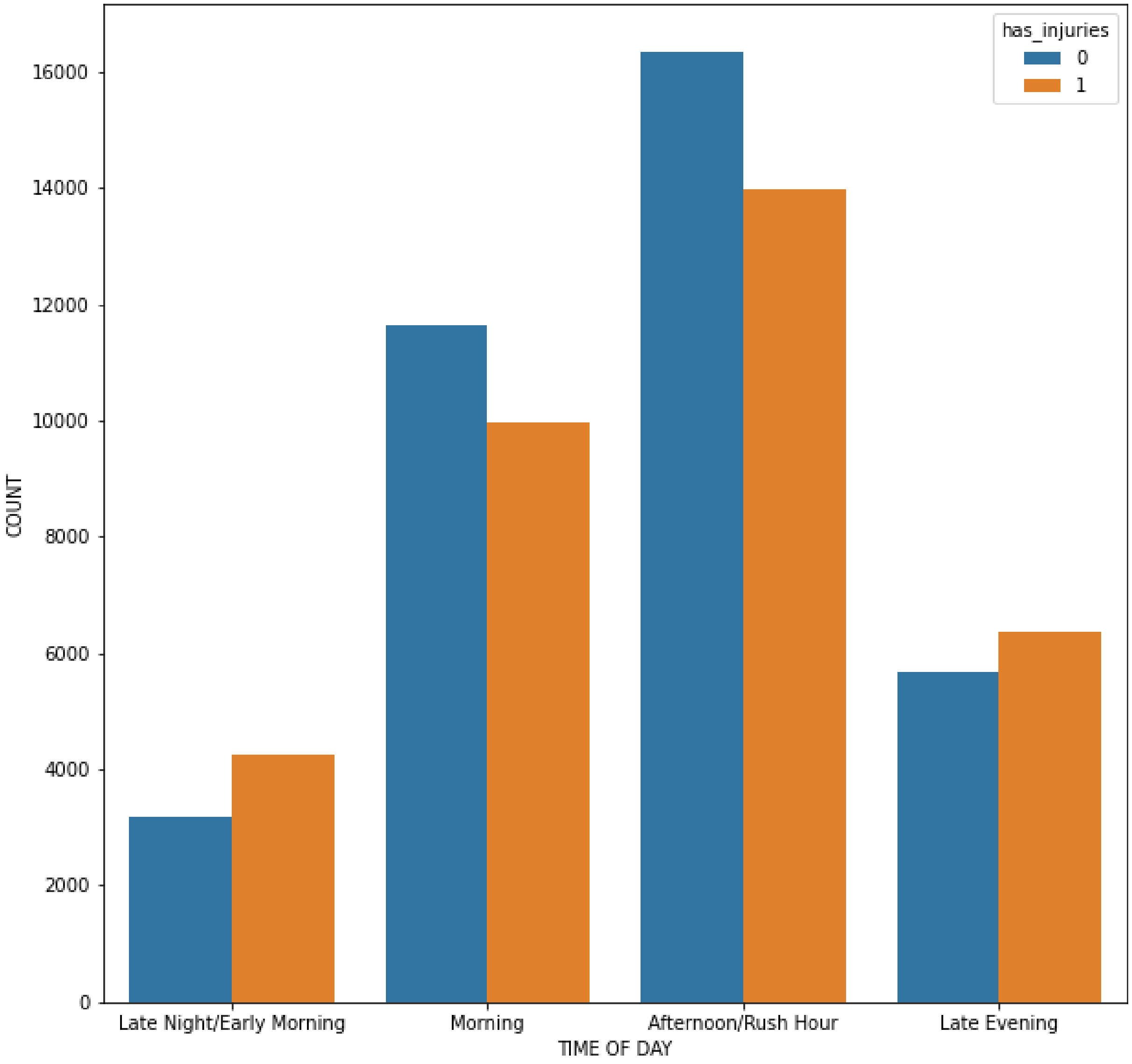
...

In [852]:

```
plt.figure(figsize=(10,10))
ax_time = sns.countplot(x="time_bins", hue="has_injuries", data=df_bal)
ax_time.set(xlabel='TIME OF DAY', ylabel='COUNT')
```

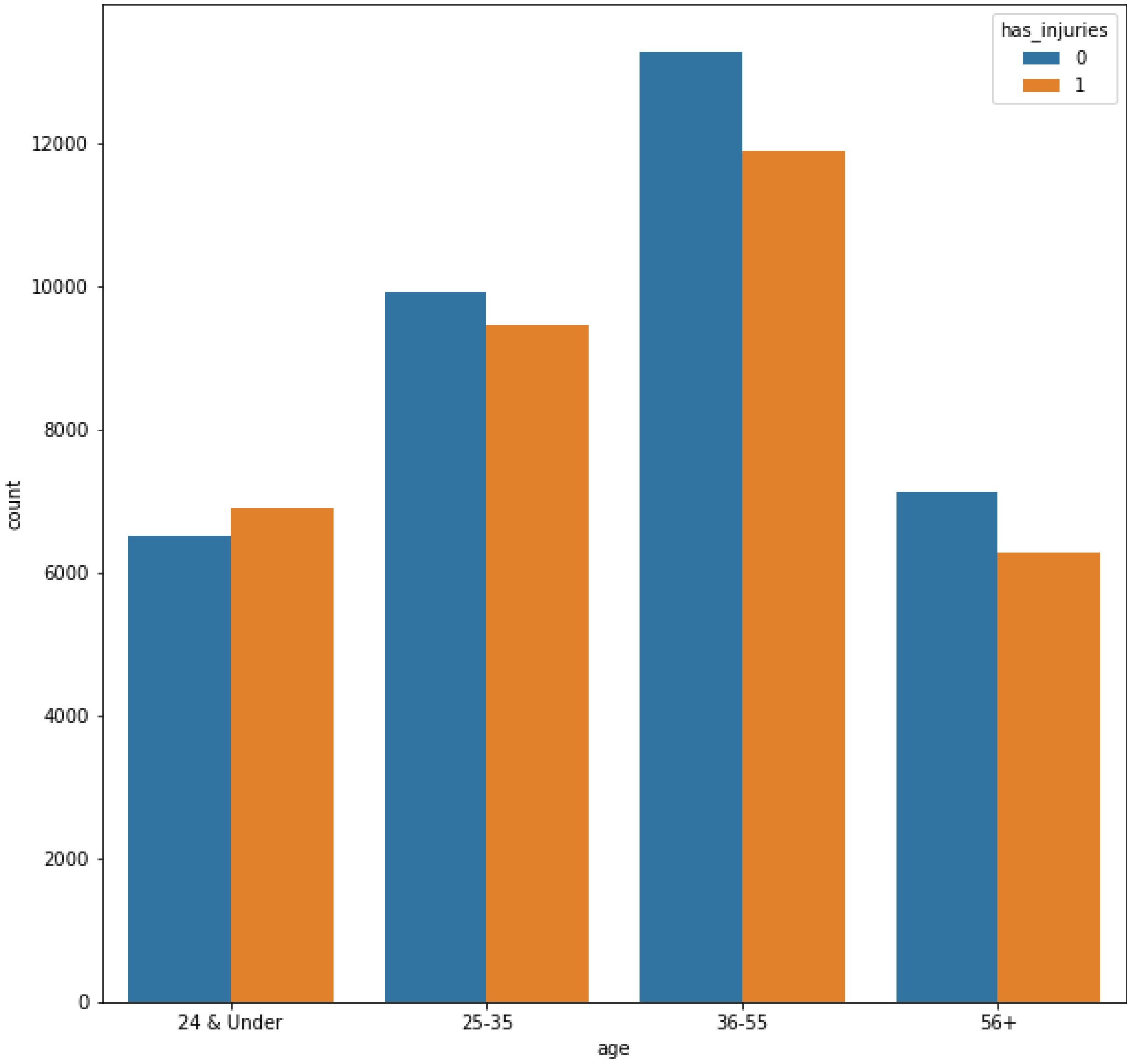
Out[852]:

```
[Text(0.5, 0, 'TIME OF DAY'), Text(0, 0.5, 'COUNT')]
```



```
In [840]: plt.figure(figsize=(10,10))
          sns.countplot(x="age", hue="has_injuries", data=df_bal)
```

```
Out[840]: <AxesSubplot:xlabel='age', ylabel='count'>
```

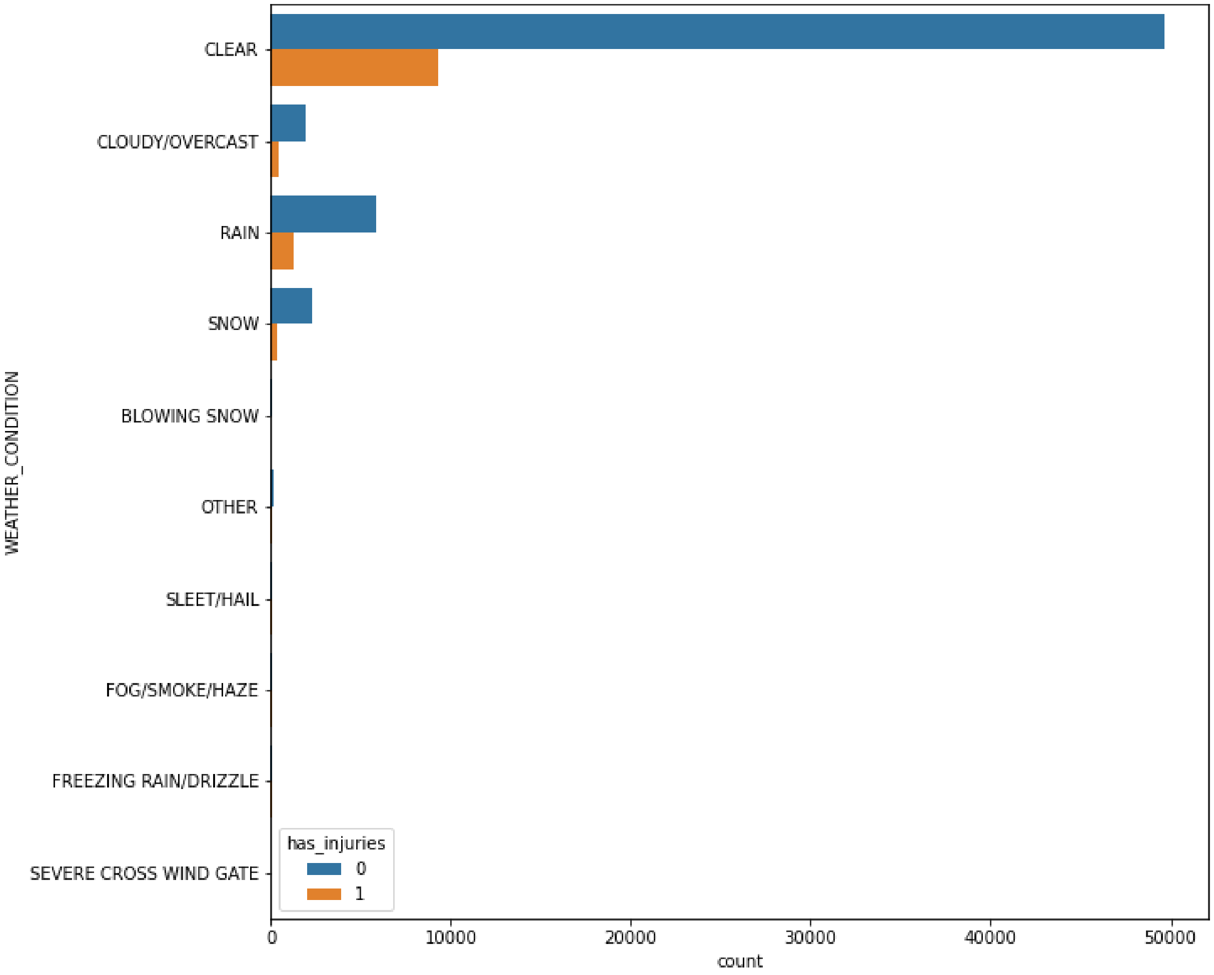


In [846]:

```
plt.figure(figsize=(10,10))
sns.countplot(y="WEATHER_CONDITION", hue="has_injuries", data=df_reduced)
```

Out[846]:

```
<AxesSubplot:xlabel='count', ylabel='WEATHER_CONDITION'>
```



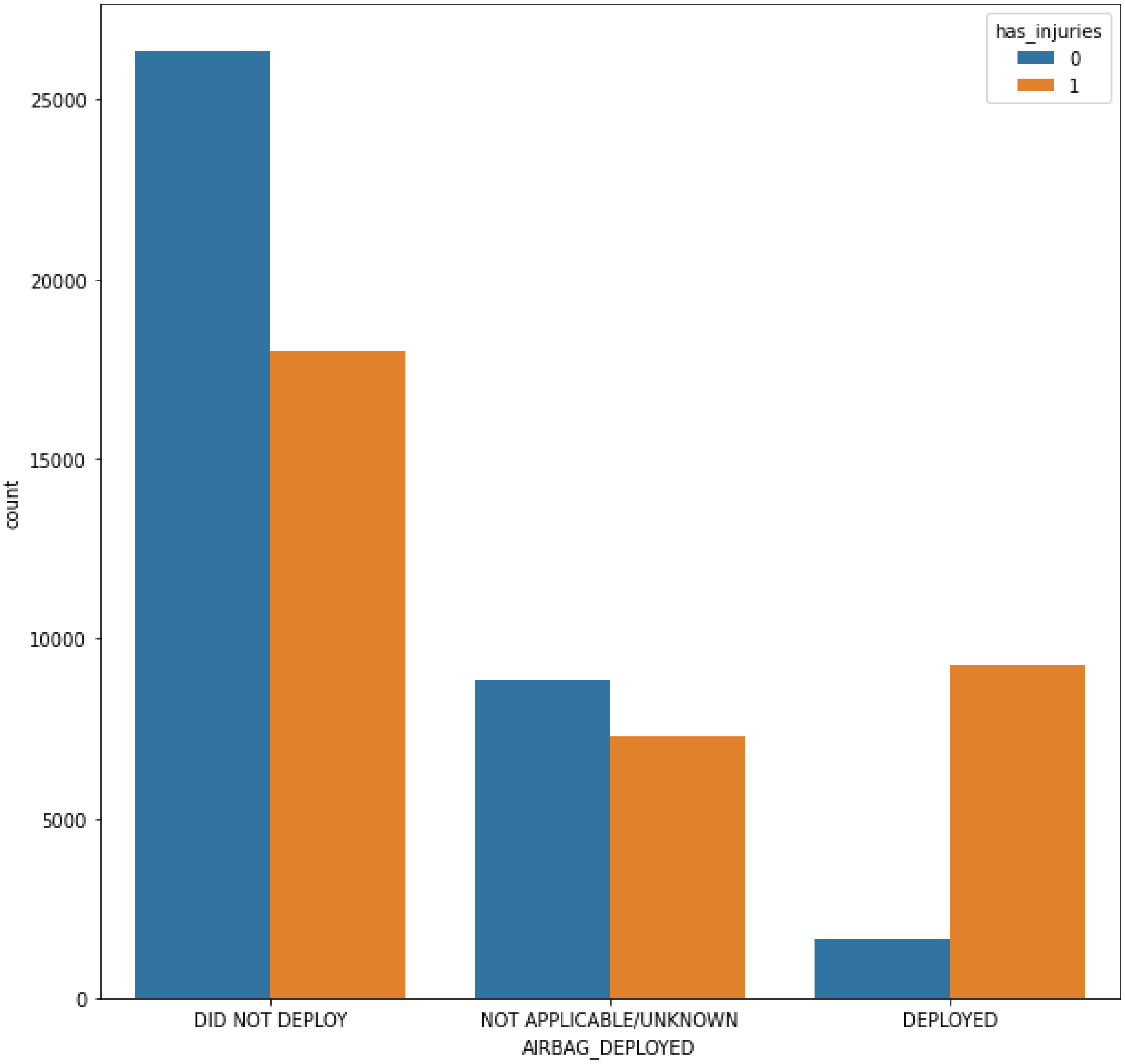
In [ ]:

In [851]:

```
plt.figure(figsize=(10,10))
sns.countplot(x="AIRBAG_DEPLOYED", hue="has_injuries", data=df_bal)
```

Out[851]:

```
<AxesSubplot:xlabel='AIRBAG_DEPLOYED', ylabel='count'>
```



In [ ]:

```
plt.figure(figsize=(10,10))
sns.countplot(x="SEX", hue="has_injuries", data=df_bal)
```

In [700]:

```
# Splitting features into numeric and categorical
numeric_columns = [
    'POSTED_SPEED_LIMIT',
#    'NUM_UNITS',
    'VEHICLE_YEAR',
#    'OCCUPANT_CNT',
    'AGE',

]

cat_columns = [
    'WEATHER_CONDITION',
    'ROADWAY_SURFACE_COND',
#    'ROAD_DEFECT',
#    'CRASH_HOUR',
    'CRASH_DAY_OF_WEEK',
    'CRASH_MONTH',
#    'ZIPCODE',
    'SEX',
    'SAFETY_EQUIPMENT',
    'AIRBAG_DEPLOYED',
    'time_bins',
    'speed_limit',
    'age',
    'vehicle_year',
#    'Airbag',

]


Target1 = df_bal[[
    'has_injuries',
#    'has_fatality',
]]

Target2 = df_bal[[
#    'has_injuries',
    'has_fatality',
]]

numeric_df = df_bal[numeric_columns]
cat_df = df_bal[cat_columns]
```

Train_Test Split

In [1020]:

```python
X = pd.concat([numeric_df,cat_df] , axis = 1)
y = Target1

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .25, r

X_train_nums = X_train [[
#    'POSTED_SPEED_LIMIT',
#    'VEHICLE_YEAR',
#    'AGE',

]]

X_train_cats = X_train [[
    'WEATHER_CONDITION',
    'ROADWAY_SURFACE_COND',
#    'ROAD_DEFECT',
#    'CRASH_HOUR',
#    'CRASH_DAY_OF_WEEK',
#    'CRASH_MONTH',
#    'ZIPCODE',
#    'SEX',
#    'SAFETY_EQUIPMENT',
#    'AIRBAG_DEPLOYED',
#    'time_bins',
    'speed_limit',
    'age',
#    'vehicle_year',
]]

X_train = pd.concat([X_train_nums,X_train_cats] , axis = 1)
```

In [449]:

```python
# Cheking the distribution of targets

y_train.value_counts(normalize=True)
```

Out[449]:

```
has_injuries
0               0.514889
1               0.485111
dtype: float64
```

Modelling

In [178]:

```python
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline as imbPipeline
from collections import Counter
```

In [179]:

```
counter = Counter(y_train)
print(counter)
print(y_train.value_counts(normalize=True))
```

```
Counter({'has_injuries': 1})
has_injuries
0              0.514778
1              0.485222
dtype: float64
```

In [1021]:

```
# One Hot Encoding categorical data


categorical_pipeline = Pipeline(steps=[
    ('ohe', OneHotEncoder(drop='first',
                          sparse=False))
])

trans = ColumnTransformer(transformers=[
#    ('numerical', numerical_pipeline, X_train_nums.columns),
    ('categorical', categorical_pipeline, X_train_cats.columns)
])

X_train_ohe = trans.fit_transform(X_train)
X_test_ohe = trans.transform(X_test)
```

```
C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\skl
earn\compose\_column_transformer.py:437: FutureWarning:

Given feature/column names or counts do not match the ones for the data given d
uring fit. This will fail from v0.24.
```

Dummy Model

In [586]:

```python
from sklearn.dummy import DummyClassifier
dummy_class = DummyClassifier(strategy = 'most_frequent')
dummy_class.fit(X_train,y_train)
#dummy_regr.predict(X_train)
y_dum = dummy_class.predict(X_test)
dummy_class.score(X_train, y_train)
dummy_class.score(X_test, y_test)
#dummy_train_RMSE = MSE(y_train,dummy_class.predict(X_train),squared = False
#dummy_test_RMSE = MSE(y_test,dummy_class.predict(X_test), squared = False)

print(classification_report(y_test, y_dum))
```

```
              precision    recall  f1-score   support

           0       0.52      1.00      0.68      9267
           1       0.00      0.00      0.00      8588

    accuracy                           0.52     17855
   macro avg       0.26      0.50      0.34     17855
weighted avg       0.27      0.52      0.35     17855


C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\skl
earn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no predicted samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

First Model

In [1022]:

```
DT = DecisionTreeClassifier(random_state = 42, class_weight='balanced', max_
DT.fit(X_train_ohe, y_train)
y_DT = DT.predict(X_test_ohe)
classification_report(y_test, y_DT)

plot_confusion_matrix(DT, X_test_ohe, y_test)
for fi, feature in zip(DT.feature_importances_, X_train.columns):
    print(fi, feature)
roc_auc_score(y_test, y_DT)

y_DT = DT.predict(X_test_ohe)

y_DT_proba = DT.predict_proba(X_test_ohe)[:, 1]
print(classification_report(y_test, y_DT))

print(DT.score(X_test_ohe, y_test))
print(DT.score(X_train_ohe, y_train))
```

```
0.03119341299074882 WEATHER_CONDITION
0.016854597790851588 ROADWAY_SURFACE_COND
0.012908849951574838 speed_limit
0.010801504264895604 age
              precision    recall  f1-score   support

           0       0.60      0.18      0.28      9267
           1       0.50      0.87      0.63      8588

    accuracy                           0.51     17855
   macro avg       0.55      0.53      0.46     17855
weighted avg       0.55      0.51      0.45     17855

0.5122934752170261
0.5241678024009111
```

0
1701
7566
7000
6000

Second Model

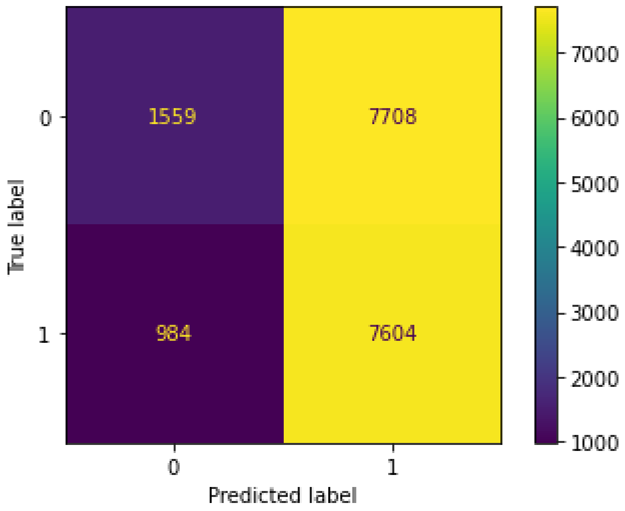
In [1023]:

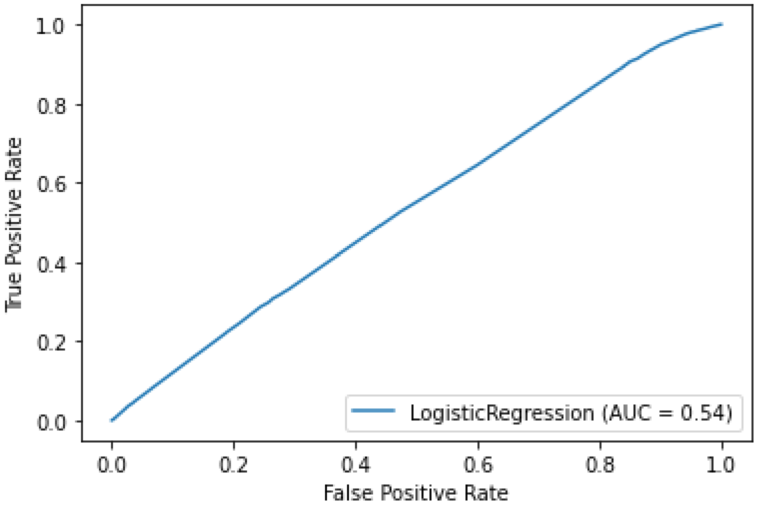
```
LR_simple = LogisticRegression(max_iter = 1e3, random_state = 42, class_weig
# Create the GridSearchCV object with different hyperparameters


LR_simple.fit(X_train_ohe, y_train)


# Predict the label with the best model
y_pred_LR_simple = LR_simple.predict_proba(X_test_ohe)[:,1]
y_pred_LR_simple_ = LR_simple.predict(X_test_ohe)

print(LR_simple.score(X_test_ohe, y_test))
print(LR_simple.score(X_train_ohe, y_train))

print(classification_report(y_test, y_pred_LR_simple_))
plot_confusion_matrix(LR_simple, X_test_ohe, y_test)
plot_roc_curve(LR_simple, X_test_ohe, y_test);
roc_auc_score(y_test, y_pred_LR_simple)
```

```
C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\skl
earn\utils\validation.py:72: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().


0.51318958274993
0.5224501988312827
              precision    recall  f1-score   support

           0       0.61      0.17      0.26      9267
           1       0.50      0.89      0.64      8588

    accuracy                           0.51     17855
   macro avg       0.55      0.53      0.45     17855
weighted avg       0.56      0.51      0.44     17855
```

Out[1023]:

```
0.5401740612011843
```




1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
LogisticRegression (AUC = 0.54)

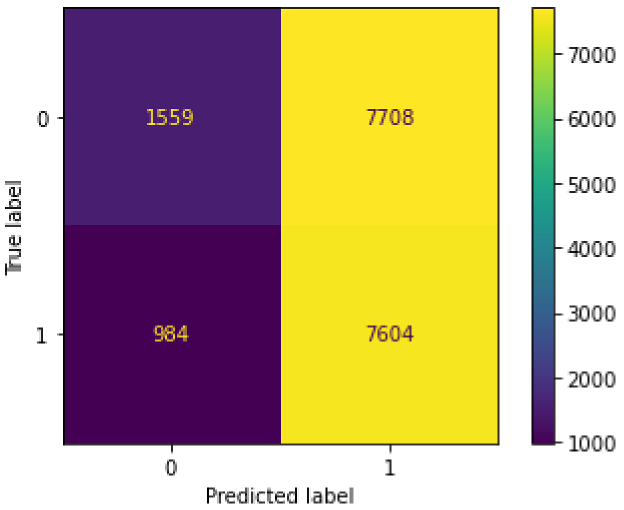
In [1015]:

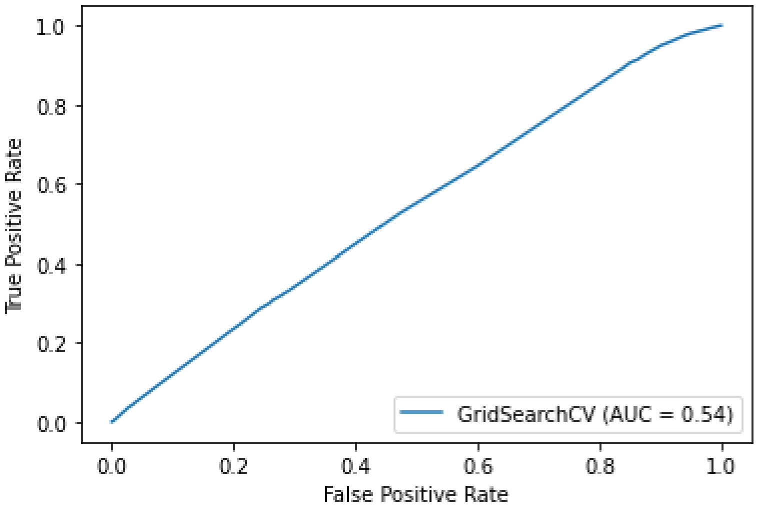
```python
# Logistic Regression Model with grid search

LR_pipe = LogisticRegression(max_iter = 1e3, random_state = 42, class_weight
# Create the GridSearchCV object with different hyperparameters
parameters_LR = {
    'C': [1, 10, 100],
    'solver': ['lbfgs','sag', 'saga']
}

cv_LR = GridSearchCV(LR_pipe, param_grid=parameters_LR, verbose = 2, n_jobs

cv_LR.fit(X_train_ohe, y_train)


# Predict the label with the best model
y_pred_LR = cv_LR.predict(X_test_ohe)
print(cv_LR.score(X_test_ohe, y_test))
print(cv_LR.score(X_train_ohe, y_train))
print(cv_LR.best_params_)
print(classification_report(y_test, y_pred_LR))
plot_confusion_matrix(cv_LR, X_test_ohe, y_test)
plot_roc_curve(cv_LR, X_test_ohe, y_test);
roc_auc_score(y_test, y_pred_LR)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    0.6s
[Parallel(n_jobs=-1)]: Done  37 out of  45 | elapsed:    1.8s remaining:    0.3
s
[Parallel(n_jobs=-1)]: Done  45 out of  45 | elapsed:    2.1s finished
C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\skl
earn\utils\validation.py:72: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().


0.51318958274993
0.5224501988312827
{'C': 1, 'solver': 'lbfgs'}
              precision    recall  f1-score   support

           0       0.61      0.17      0.26      9267
           1       0.50      0.89      0.64      8588

    accuracy                           0.51     17855
   macro avg       0.55      0.53      0.45     17855
weighted avg       0.56      0.51      0.44     17855
```

Out[1015]:

```
0.526826438490994
```


1559
7708
984
7604
True label
0
1
Predicted label
0
1
7000
6000
5000
4000
3000
2000
1000


True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
GridSearchCV (AUC = 0.54)

In [1012]:

```
# DecisionTree Model with Pipeline
DT = DecisionTreeClassifier(random_state = 42, class_weight='balanced')

parameters_DT = {
    'max_depth': [6,15,20],
    'min_samples_split': [10,15],
    'criterion': ['entropy','gini']
}

cv_DT = GridSearchCV(DT, param_grid=parameters_DT, verbose = 2, n_jobs = -1,

cv_DT.fit(X_train_ohe, y_train)

# Predict the label with the best model
y_pred_DT = cv_DT.predict(X_test_ohe)
print(cv_DT.score(X_test_ohe, y_test))
print(cv_DT.score(X_train_ohe, y_train))
print(cv_DT.best_params_)
print(classification_report(y_test, y_pred_DT))
plot_confusion_matrix(cv_DT, X_test_ohe, y_test)
plot_roc_curve(cv_DT, X_test_ohe, y_test);
roc_auc_score(y_test, y_pred_LR)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed:    0.5s remaining:
0.0s
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed:    0.5s finished

0.5397563191433723
0.5509891530870029
{'criterion': 'gini', 'max_depth': 6, 'min_samples_split': 15}
              precision    recall  f1-score   support

           0       0.60      0.17      0.27      9267
           1       0.50      0.88      0.63      8588

    accuracy                           0.51     17855
   macro avg       0.55      0.52      0.45     17855
weighted avg       0.55      0.51      0.44     17855
```

KNN Model

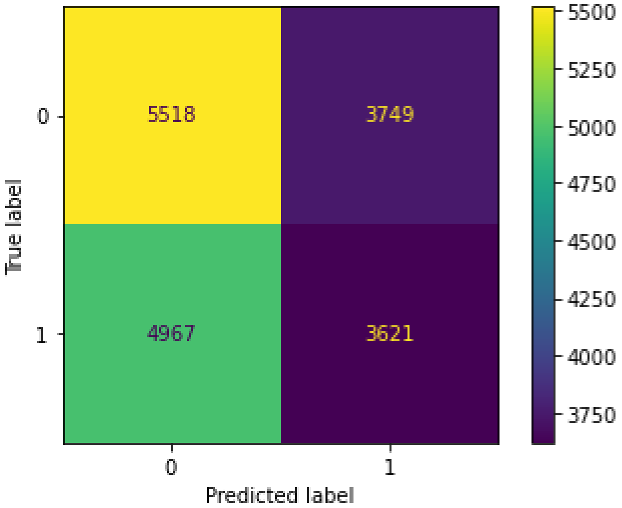
In [1011]:

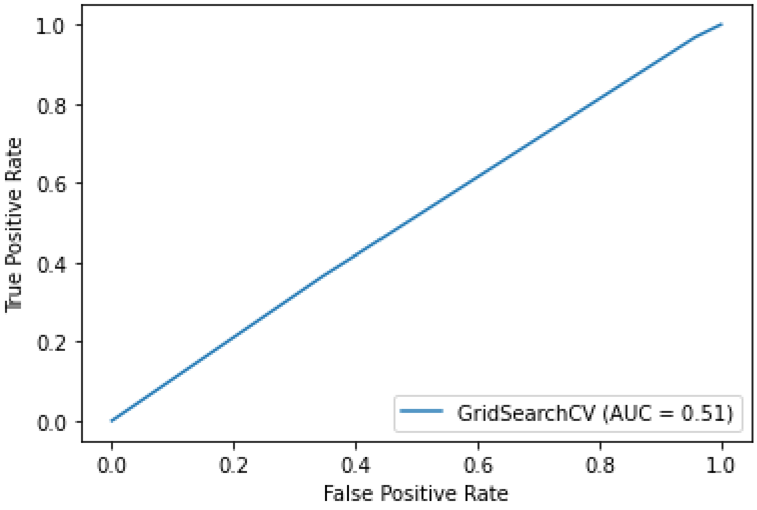
```python
# KNN Model with Pipeline
KNN = KNeighborsClassifier()

# Create the GridSearchCV object with different hyperparameters
parameters_KNN = {
    'n_neighbors': [3, 5],
    'metric': ['minkowski', 'manhattan'],
    'weights': ['uniform'],
}

cv_KNN = GridSearchCV(KNN, param_grid=parameters_KNN, verbose = 2, n_jobs =

cv_KNN.fit(X_train_ohe, y_train)

# Predict the label with the best model
y_pred_KNN = cv_KNN.predict(X_test_ohe)
print(cv_KNN.score(X_test_ohe, y_test))
print(cv_KNN.score(X_train_ohe, y_train))
print(cv_KNN.best_params_)
print(classification_report(y_test, y_pred_KNN))
plot_confusion_matrix(cv_KNN, X_test_ohe, y_test)
plot_roc_curve(cv_KNN, X_test_ohe, y_test);
roc_auc_score(y_test, y_pred_LR)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done  11 out of  20 | elapsed:   33.6s remaining:   27.5
s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:   40.1s finished
C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\skl
earn\model_selection\_search.py:765: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().




0.5121193132936767
0.5209096479133745
{'metric': 'minkowski', 'n_neighbors': 5, 'weights': 'uniform'}
              precision    recall  f1-score   support

           0       0.53      0.60      0.56      9267
           1       0.49      0.42      0.45      8588

    accuracy                           0.51     17855
   macro avg       0.51      0.51      0.51     17855
weighted avg       0.51      0.51      0.51     17855
```

Out[1011]:

```
0.526826438490994
```





Stacked Model

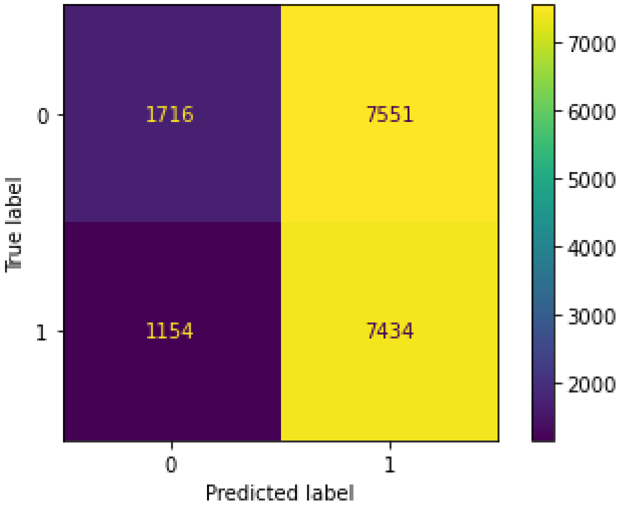
In [1010]:

```python
LR_Stack = LogisticRegression(max_iter = 1e3, random_state = 42, class_weigh
DT_Stack = DecisionTreeClassifier(random_state = 42, class_weight='balanced'
KNN_Stack = KNeighborsClassifier(n_neighbors=5, metric ='minkowski', weights

avg = VotingClassifier(estimators=[
    ('LR', LR_Stack),
    ('KNN', KNN_Stack),
    ('DT', DT_Stack)])
avg.fit(X_train_ohe, y_train)

y_pred_avg = avg.predict(X_test_ohe)
print(avg.score(X_test_ohe, y_test))
print(avg.score(X_train_ohe, y_train))
print(classification_report(y_test, y_pred_avg))
plot_confusion_matrix(avg, X_test_ohe, y_test)
#plot_roc_curve(avg, X_test_ohe, y_test);
roc_auc_score(y_test, y_pred_avg)

```

```
C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\skl
earn\utils\validation.py:72: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().


0.5124614953794455
0.5237010623004686
              precision    recall  f1-score   support

           0       0.60      0.19      0.28      9267
           1       0.50      0.87      0.63      8588

    accuracy                           0.51     17855
   macro avg       0.55      0.53      0.46     17855
weighted avg       0.55      0.51      0.45     17855
```

Out[1010]:

```
0.5253998253640674
```



XGBoost Model

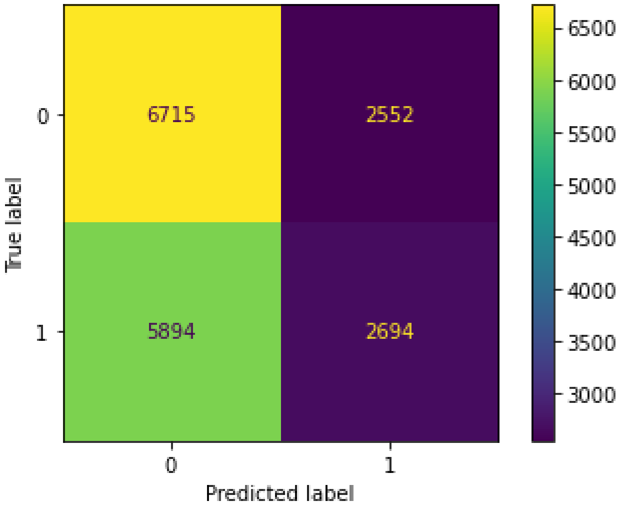
In [1005]:

```python
boost = xgboost.XGBClassifier(n_estimators = 1000, random_state=42, learning

boost.fit(X_train_ohe, y_train)

y_pred_boost = boost.predict(X_test_ohe)
print(boost.score(X_test_ohe, y_test))
print(boost.score(X_train_ohe, y_train))
print(classification_report(y_test, y_pred_boost))
plot_confusion_matrix(boost, X_test_ohe, y_test)
#plot_roc_curve(avg, X_test_ohe, y_test);
roc_auc_score(y_test, y_pred_boost)

```

```
C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\skl
earn\utils\validation.py:72: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().


0.5269672360683282
0.5337266396579728
              precision    recall  f1-score   support

           0       0.53      0.72      0.61      9267
           1       0.51      0.31      0.39      8588

    accuracy                           0.53     17855
   macro avg       0.52      0.52      0.50     17855
weighted avg       0.52      0.53      0.51     17855
```

Out[1005]:

```
0.5191538741800025
```



Random Forrest

In [1009]:

```python
# Random Forrest Model with Pipeline
RF_pipe = imbPipeline(steps=[
    ('RF', RandomForestClassifier(random_state = 42, class_weight='balanced'
])

# Create the GridSearchCV object with different hyperparameters
parameters_RF = {
    'RF__n_estimators': [100],
    'RF__max_depth': [6, 8,10,20],
    'RF__min_samples_split': [10, 15],
    'RF__criterion': ['gini']
}

cv_RF = GridSearchCV(RF_pipe, param_grid=parameters_RF, verbose = 2, n_jobs

cv_RF.fit(X_train_ohe, y_train)

# Predict the label with the best model
y_pred_RF = cv_RF.predict_proba(X_test_ohe)
print(cv_RF.score(X_test_ohe, y_test))
print(cv_RF.score(X_train_ohe, y_train))
print(cv_RF.best_params_)
#print(classification_report(y_test, y_pred_RF))
plot_confusion_matrix(cv_RF, X_test_ohe, y_test)
#roc_auc_score(y_test, y_pred_RF)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    2.3s
[Parallel(n_jobs=-1)]: Done  30 out of  40 | elapsed:    5.1s remaining:
1.6s
[Parallel(n_jobs=-1)]: Done  40 out of  40 | elapsed:    6.4s finished
C:\Users\milad\Documents\Flatiron\Anaconda\envs\learn-env\lib\site-packages\i
mblearn\pipeline.py:281: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().


0.5355559042812542
0.550346051901595
{'RF__criterion': 'gini', 'RF__max_depth': 6, 'RF__min_samples_split': 15, 'R
F__n_estimators': 100}
```

Out[1009]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x26704ed80
```