In [56]:

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

In [57]:

```
filepath = "/resources/labs/coursera/ML0101EN/Crash_Analysis_System__CAS__Data.csv"
df_test = pd.read_csv(filepath)
```

In [58]:

```
df_test.shape
```

Out[58]:

```
(16338, 71)
```

In [59]:

```
df_test.columns
```

Out[59]:

```
Index(['X', 'Y', 'OBJECTID', 'advisorySpeed', 'areaUnitID', 'bicycle',
       'bridge', 'bus', 'carStationWagon', 'cliffBank',
       'crashDirectionDescription', 'crashLocation1', 'crashLocation2',
       'crashRoadSideRoad', 'crashSeverity', 'crashSHDescription', 'crashY
ear',
       'debris', 'directionRoleDescription', 'ditch', 'fatalCount', 'fenc
e',
       'flatHill', 'guardRail', 'holiday', 'houseOrBuilding', 'intersectio
n',
       'kerb', 'light', 'meshblockId', 'minorInjuryCount', 'moped',
       'motorcycle', 'NumberOfLanes', 'objectThrownOrDropped', 'otherObjec
t',
       'otherVehicleType', 'overBank', 'parkedVehicle', 'pedestrian',
       'phoneBoxEtc', 'postOrPole', 'region', 'roadCharacter', 'roadLane',
       'roadSurface', 'roadworks', 'schoolBus', 'seriousInjuryCount',
       'slipOrFlood', 'speedLimit', 'strayAnimal', 'streetLight', 'suv',
       'taxi', 'temporarySpeedLimit', 'tlaId', 'tlaName', 'trafficContro
l',
       'trafficIsland', 'trafficSign', 'train', 'tree', 'truck',
       'unknownVehicleType', 'urban', 'vanOrUtility', 'vehicle', 'waterRiv
er',
       'weatherA', 'weatherB'],
      dtype='object')
```

In [60]:

```
indexNames = df_test[df_test['crashSeverity'] =='Non-Injury Crash'].index
df_test.drop(indexNames, inplace=True)
```

In [61]:

```
df_test.shape
```

Out[61]:

```
(5537, 71)
```

In [62]:

```
df = df_test[['crashSeverity','fatalCount', 'minorInjuryCount', 'seriousInjuryCount','c
rashYear','region','speedLimit','weatherA', 'weatherB']]
df_map = df_test[['crashSeverity','region','X','Y']]
```

In [63]:

```
df.shape
```

Out[63]:

```
(5537, 9)
```

In [64]:

```
df.info
```

Out[64]:

```
<bound method DataFrame.info of        crashSeverity  fatalCount  minorInj
uryCount  seriousInjuryCount  \
1      Serious Crash         0.0              1.0               1.0
3      Serious Crash         0.0              0.0               1.0
5        Minor Crash         0.0              1.0               0.0
7      Serious Crash         0.0              0.0               1.0
11       Minor Crash         0.0              1.0               0.0
...              ...         ...              ...               ...
16329    Minor Crash         0.0              1.0               0.0
16330    Minor Crash         0.0              7.0               0.0
16333    Minor Crash         0.0              3.0               0.0
16334    Minor Crash         0.0              1.0               0.0
16336  Serious Crash         0.0              0.0               1.0

       crashYear                    region  speedLimit    weatherA  \
1           2017          Southland Region       100.0  Light rain
3           2016          Northland Region       100.0        Fine
5           2018        Wellington Region        50.0        Fine
7           2016  Manawatu-Wanganui Region        70.0        Fine
11          2006        Canterbury Region        50.0        Fine
...          ...                       ...         ...         ...
16329       2014          Auckland Region        50.0  Heavy rain
16330       2007             Otago Region       100.0        Fine
16333       2018        Canterbury Region       100.0        Fine
16334       2013  Manawatu-Wanganui Region       100.0        Fine
16336       2016             Otago Region       100.0        Fine

         weatherB
1      Strong wind
3      Strong wind
5      Strong wind
7            Frost
11     Strong wind
...            ...
16329  Strong wind
16330  Strong wind
16333  Strong wind
16334        Frost
16336        Frost

[5537 rows x 9 columns]>
```

In [65]:

```
from numpy import *
import warnings
warnings.filterwarnings('ignore')
df.replace("?", np.nan, inplace = True)
```

In [66]:

```
missing_data = df.isnull()
missing_data.head(5)
```

Out[66]:

	crashSeverity	fatalCount	minorInjuryCount	seriousInjuryCount	crashYear	region	speed
1	False	False	False	False	False	False	
3	False	False	False	False	False	False	
5	False	False	False	False	False	False	
7	False	False	False	False	False	False	
11	False	False	False	False	False	False	

In [67]:

```
missing_data = df.isnull()
for column in missing_data.columns.values.tolist():
    print(column)
    print(missing_data[column].value_counts())
    print("---------------------------------")
```

```
crashSeverity
False    5537
Name: crashSeverity, dtype: int64
---------------------------------
fatalCount
False    5537
Name: fatalCount, dtype: int64
---------------------------------
minorInjuryCount
False    5537
Name: minorInjuryCount, dtype: int64
---------------------------------
seriousInjuryCount
False    5537
Name: seriousInjuryCount, dtype: int64
---------------------------------
crashYear
False    5537
Name: crashYear, dtype: int64
---------------------------------
region
False    5537
Name: region, dtype: int64
---------------------------------
speedLimit
False    5536
True        1
Name: speedLimit, dtype: int64
---------------------------------
weatherA
False    5537
Name: weatherA, dtype: int64
---------------------------------
weatherB
False    5537
Name: weatherB, dtype: int64
---------------------------------
```

In [68]:

```
df['crashSeverity'].value_counts()
```

Out[68]:

```
Minor Crash      4234
Serious Crash    1146
Fatal Crash       157
Name: crashSeverity, dtype: int64
```

In [69]:

```
df["crashSeverity"].replace("Minor Crash", '1', inplace=True)
df["crashSeverity"].replace("Serious Crash", '2', inplace=True)
df["crashSeverity"].replace("Fatal Crash", '3', inplace=True)
df["crashSeverity"].value_counts()
```

Out[69]:

```
1    4234
2    1146
3     157
Name: crashSeverity, dtype: int64
```

In [25]:

```
df["speedLimit"].replace(np.nan, "N", inplace=True)
df["speedLimit"].describe()
```

Out[25]:

```
count     5537.0
unique      11.0
top        100.0
freq      3162.0
Name: speedLimit, dtype: float64
```

In [26]:

```
df.head(5)
```

Out[26]:

	crashSeverity	fatalCount	minorInjuryCount	seriousInjuryCount	crashYear	region	sl
1	2	0.0	1.0	1.0	2017	Southland Region	
3	2	0.0	0.0	1.0	2016	Northland Region	
5	1	0.0	1.0	0.0	2018	Wellington Region	
7	2	0.0	0.0	1.0	2016	Manawatu-Wanganui Region	
11	1	0.0	1.0	0.0	2006	Canterbury Region	

In [27]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5537 entries, 1 to 16336
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   crashSeverity       5537 non-null   object 
 1   fatalCount          5537 non-null   float64
 2   minorInjuryCount    5537 non-null   float64
 3   seriousInjuryCount  5537 non-null   float64
 4   crashYear           5537 non-null   int64  
 5   region              5537 non-null   object 
 6   speedLimit          5537 non-null   object 
 7   weatherA            5537 non-null   object 
 8   weatherB            5537 non-null   object 
dtypes: float64(3), int64(1), object(5)
memory usage: 592.6+ KB
```

In [28]:

```
df.replace('?', np.NaN)
```

Out[28]:

	crashSeverity	fatalCount	minorInjuryCount	seriousInjuryCount	crashYear	region
1	2	0.0	1.0	1.0	2017	Southland Region
3	2	0.0	0.0	1.0	2016	Northland Region
5	1	0.0	1.0	0.0	2018	Wellington Region
7	2	0.0	0.0	1.0	2016	Manawatu-Wanganui Region
11	1	0.0	1.0	0.0	2006	Canterbury Region
...	...	...	...	...	...	...
16329	1	0.0	1.0	0.0	2014	Auckland Region
16330	1	0.0	7.0	0.0	2007	Otago Region
16333	1	0.0	3.0	0.0	2018	Canterbury Region
16334	1	0.0	1.0	0.0	2013	Manawatu-Wanganui Region
16336	2	0.0	0.0	1.0	2016	Otago Region

5537 rows × 9 columns

In [70]:

```
df_map.dropna(subset=["X"], axis=0, inplace=True)
df_map.shape
```

Out[70]:

```
(5537, 4)
```

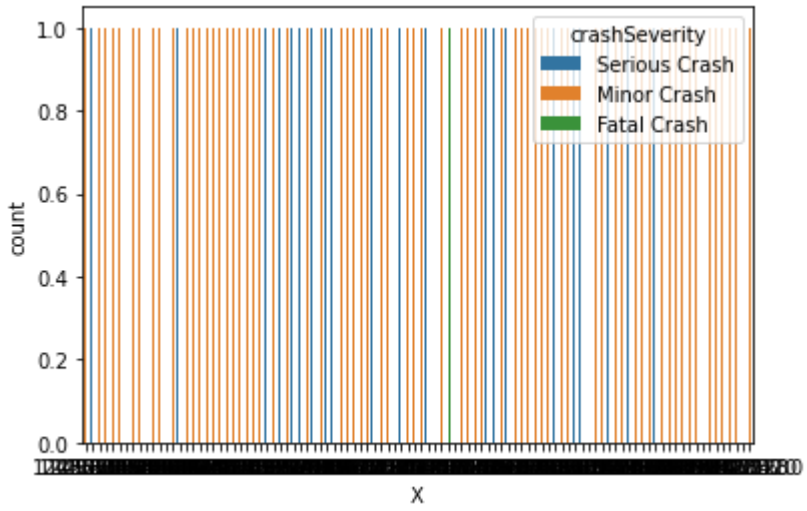
In [49]:

```
"""

seaborn.countplot(x=None, y=None, hue=None,
                  data=None, order=None, hue_order=None,
                  orient=None, color=None, palette=None,
                  saturation=0.75, dodge=True, ax=None, **kwargs)
"""
sns.countplot(x= "X", data= df_map.head(100), hue="crashSeverity")
```

Out[49]:

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



In [71]:

```
bins = np.linspace(min(df_map["X"]), max(df_map["X"]), 4)
group_names = ['Low', 'Medium', 'High']
df_map['X-binned'] = pd.cut(df_map['X'], bins, labels=group_names, include_lowest=True)
df_map[['X','X-binned']].head(10)
```

Out[71]:

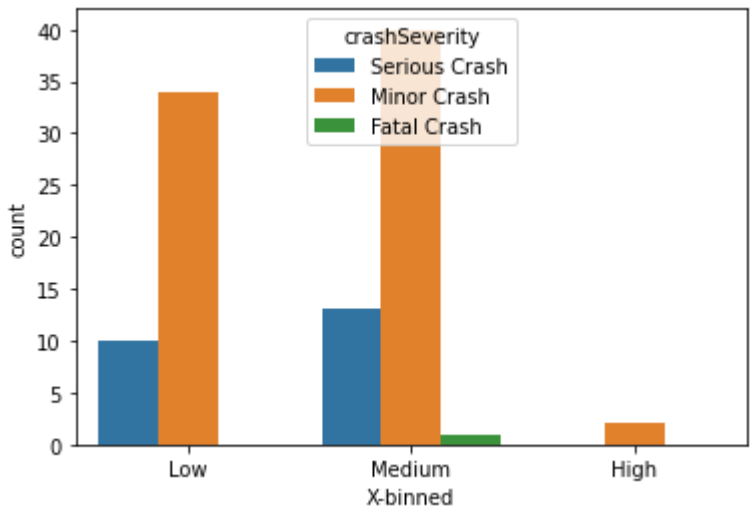
	X	X-binned
1	1249628.0	Low
3	1737511.0	Medium
5	1750808.0	Medium
7	1823595.0	Medium
11	1567471.0	Low
13	1831689.0	Medium
15	2038842.0	High
18	1765486.0	Medium
20	1599881.0	Low
22	1918276.0	Medium

In [51]:

```
sns.countplot(x = "X-binned", data=df_map.head(100), hue="crashSeverity")
```

Out[51]:

```
<AxesSubplot:xlabel='X-binned', ylabel='count'>
```



In [72]:

```
Feature = df[['region', 'weatherA', 'weatherB']]
X = Feature.values
X[0:5]
```

Out[72]:

```
array([['Southland Region', 'Light rain', 'Strong wind'],
       ['Northland Region', 'Fine', 'Strong wind'],
       ['Wellington Region', 'Fine', 'Strong wind'],
       ['Manawatu-Wanganui Region', 'Fine', 'Frost'],
       ['Canterbury Region', 'Fine', 'Strong wind']], dtype=object)
```

In [32]:

```
categorical_feature_mask = Feature.dtypes==object
categorical_feature_mask
```

Out[32]:

```
region      True
weatherA    True
weatherB    True
dtype: bool
```

In [81]:

```
categorical_cols = Feature.columns[categorical_feature_mask].tolist()
categorical_cols
```

Out[81]:

```
['region', 'weatherA', 'weatherB']
```

In [82]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [83]:

```
Feature[categorical_cols] = Feature[categorical_cols].apply(lambda col: le.fit_transform(col))
Feature[categorical_cols].head(10)
```

Out[83]:

	region	weatherA	weatherB
1	10	3	1
3	8	0	1
5	14	0	1
7	5	0	0
11	2	0	1
13	5	0	1
15	3	0	0
18	0	0	1
20	2	2	1
22	4	0	0

In [84]:

```
X = Feature.values
X[0:5]
```

Out[84]:

```
array([[10,  3,  1],
       [ 8,  0,  1],
       [14,  0,  1],
       [ 5,  0,  0],
       [ 2,  0,  1]])
```

In [85]:

```
y =df['crashSeverity'].values
y[0:5]
```

Out[85]:

```
array(['2', '2', '1', '2', '1'], dtype=object)
```

In [86]:

```
import itertools
import numpy as np
from matplotlib.ticker import NullFormatter
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
from sklearn import preprocessing
%matplotlib inline
```

In [87]:

```
from sklearn.model_selection import train_test_split
X_train_raw, X_test, y_train_raw, y_test = train_test_split(X, y, test_size=0.4, random
_state=4)
print ('Train set:', X_train_raw.shape, y_train_raw.shape)
print ('Test set:', X_test.shape, y_test.shape)
```

```
Train set: (3322, 3) (3322,)
Test set: (2215, 3) (2215,)
```

In [104]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
```

In [92]:

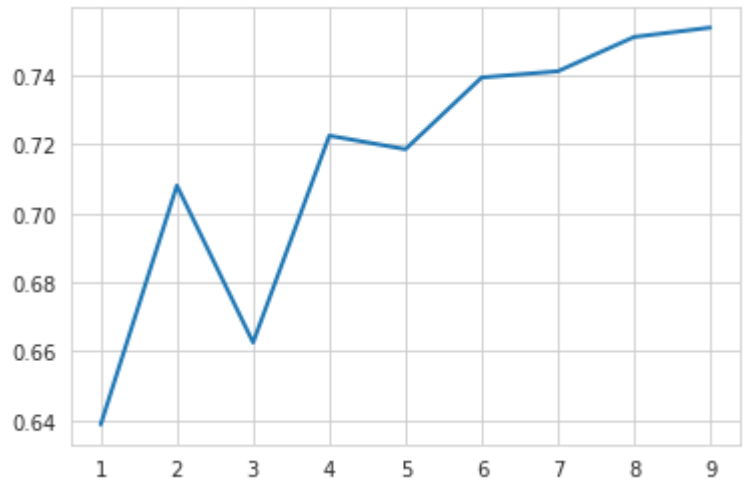
```
grid_params = {'n_neighbors': [i for i in range(1, 10)]}
grid = GridSearchCV(KNeighborsClassifier(),grid_params,cv = 5)
grid_results = grid.fit(X_train_raw, y_train_raw)
```

In [94]:

```
sns.set_style("whitegrid")
sns.lineplot(grid_params['n_neighbors'], grid_results.cv_results_['mean_test_score'],pa
lette="hls", linewidth=2)
```

Out[94]:

```
<AxesSubplot:>
```



In [95]:

```
print("The best n_neighbors was : ", grid_results.best_params_['n_neighbors'])
print("The best accuracy with:", grid_results.best_score_.round(2))
```

```
The best n_neighbors was :  9
The best accuracy with: 0.75
```

In [107]:

```
neigh = KNeighborsClassifier(n_neighbors = 6).fit(X_train_raw, y_train_raw)
yhat_train = neigh.predict(X_train_raw)
```

In [110]:

```
from sklearn.metrics import jaccard_score
```

```
---------------------------------------------------------------------------
-
ImportError                               Traceback (most recent call las
t)
<ipython-input-110-88266d8c06b1> in <module>
----> 1 from sklearn.metrics import jaccard_score

ImportError: cannot import name 'jaccard_score'
```

In [111]:

```
print ("KNN Accuracy         :{:.2f}".format(metrics.accuracy_score(y_train_raw, yhat_t
rain)))

print("KNN F1-score          :{:.2f}".format(metrics.f1_score(y_train_raw, yhat_train,
average='weighted')))
```

```
KNN Accuracy         :0.75
KNN F1-score          :0.66
```

In [ ]:

```

```