Libraries below are needed to complete this analysis

```
# Import Libraries Required
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns
```

Map to the Google drive and load the data.

```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.m
/gdrive
```

Use the same file as in the EDA.

```
accident_data = pd.read_csv('/gdrive/My Drive/VERMONT.csv',encoding_errors='ignor
```

Filter the dataset to your respective State Assignment.

```
df=accident_data.loc[accident_data['STATENAME']=="Vermont"]
#df.loc[df['column_name'] == some_value]
df
```

	STATE	STATENAME	ST_CASE	VE_TOTAL	VE_FORMS	PVH_INVL	PEDS	PERSONS	PERM
0	50	Vermont	500001	1	1	0	0	1	
1	50	Vermont	500002	2	2	0	0	4	
2	50	Vermont	500003	1	1	0	0	1	
3	50	Vermont	500004	2	2	0	0	3	

3	50	Vermont	500004	2	2	0	0	3
4	50	Vermont	500005	1	1	0	1	1
5	50	Vermont	500006	1	1	0	0	3
6	50	Vermont	500007	2	2	0	0	2
7	50	Vermont	500008	1	1	0	0	1
8	50	Vermont	500009	1	1	0	0	1
9	50	Vermont	500010	2	2	0	0	3
10	50	Vermont	500011	1	1	0	0	1
11	50	Vermont	500012	1	1	0	0	1
12	50	Vermont	500013	2	2	0	0	2
13	50	Vermont	500014	1	1	0	0	2
14	50	Vermont	500015	2	2	0	0	4
15	50	Vermont	500016	1	1	0	0	1
16	50	Vermont	500017	1	1	0	0	2
17	50	Vermont	500018	2	2	0	0	3
18	50	Vermont	500019	1	1	0	1	1
19	50	Vermont	500020	2	2	0	0	3
20	50	Vermont	500021	1	1	0	0	1

20	50	Vermont	500021	1	1	0	0	1
21	50	Vermont	500022	1	1	0	0	2
22	50	Vermont	500023	2	2	0	0	3
23	50	Vermont	500024	1	1	0	0	1
24	50	Vermont	500025	1	1	0	1	1
25	50	Vermont	500026	2	1	1	0	1
26	50	Vermont	500027	2	2	0	0	2
27	50	Vermont	500028	2	2	0	0	3
28	50	Vermont	500029	1	1	0	0	1
29	50	Vermont	500030	3	3	0	0	5
30	50	Vermont	500031	3	3	0	0	3
31	50	Vermont	500032	2	2	0	0	3
32	50	Vermont	500033	4	3	1	0	3
33	50	Vermont	500034	1	1	0	0	2
34	50	Vermont	500035	1	1	0	0	1
35	50	Vermont	500036	2	2	0	0	2
36	50	Vermont	500037	2	2	0	0	2
37	50	Vermont	500038	3	3	0	0	3
38	50	Vermont	500039	1	1	0	0	1
39	50	Vermont	500040	2	2	0	0	3
40	50	Vermont	500041	1	1	0	0	1

41	50	Vermont	500042	1	1	0	0	1
42	50	Vermont	500043	1	1	0	0	1
43	50	Vermont	500044	2	2	0	0	2

44 rows × 91 columns

Use FATALS Column as the Dependent Variable in the Classfication Analysis.

```
Y=df['FATALS'].copy()
Y
```

	FATALS
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	2
13	1

14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	2
23	1
24	1
25	1
26	1
27	2
28	1
29	1
30	1
31	1
32	1
33	1
34	1
35	1
36	1
37	1
38	1
39	1
40	1
41	1

42	1
43	1

dtype: int64

Assign Columns to the Independent variables you select to analyze.

Your challenge in this exercise is to find the columns that have the BEST Classification performance, and then to find the columns that produce the worst Classification performance.

```
X=df[['PERSONS','HARM_EV','MAN_COLL']].copy()
X
```

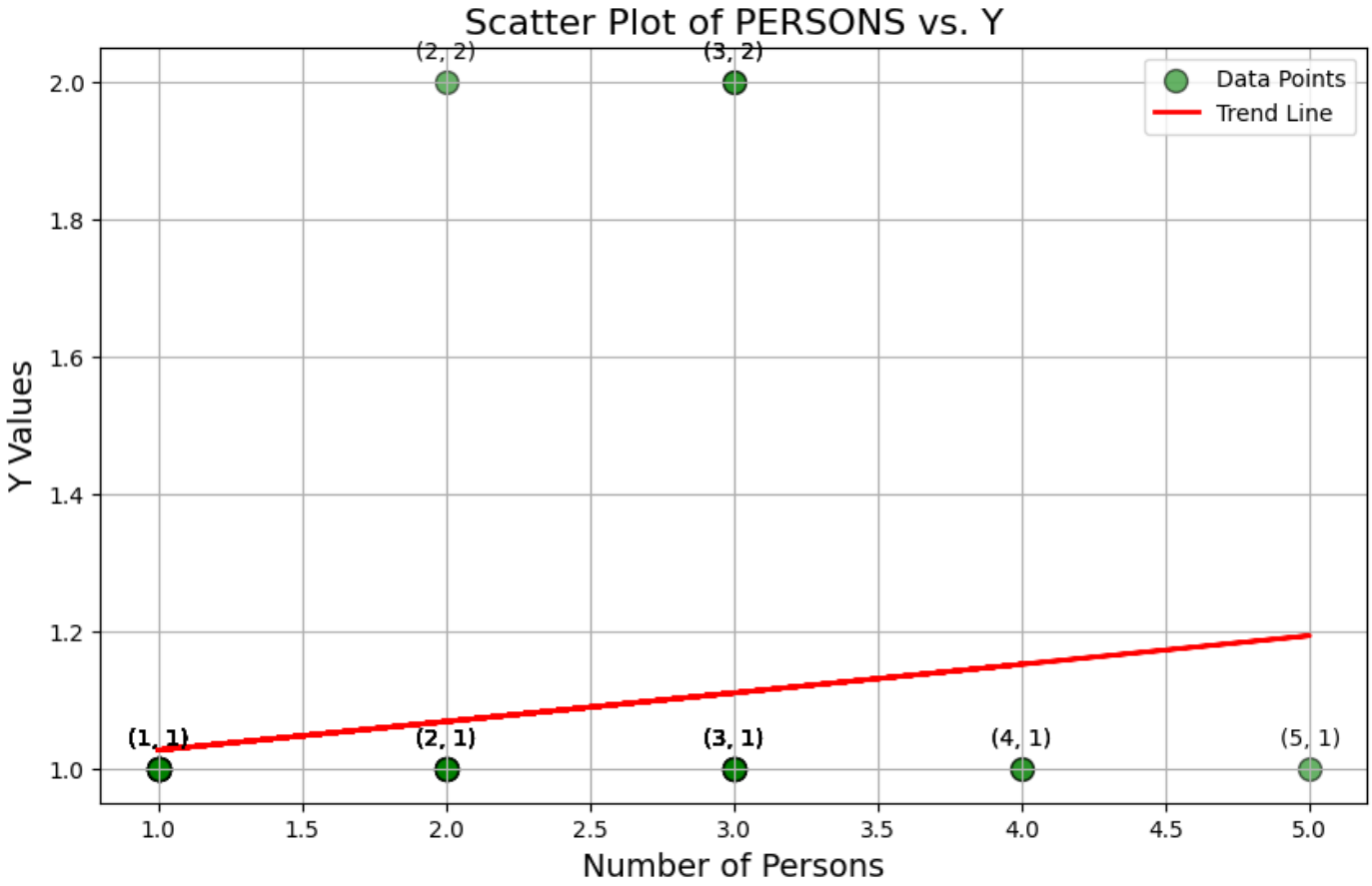
	PERSONS	HARM_EV	MAN_COLL
0	1	48	0
1	4	12	6
2	1	30	0
3	3	12	2
4	1	8	0
5	3	42	0
6	2	12	6
7	1	1	0
8	1	35	0
9	3	12	2
10	1	39	0
11	1	42	0
12	2	12	2
13	2	42	0
14	4	12	1
15	1	1	0
16	2	42	0

17	3	12	2
18	1	8	0
19	3	12	2
20	1	31	0
21	2	1	0
22	3	12	2
23	1	35	0
24	1	8	0
25	1	45	0
26	2	12	6
27	3	12	6
28	1	1	0
29	5	12	8
30	3	12	6
31	3	12	2
32	3	12	1
33	2	42	0
34	1	42	0
35	2	12	6
36	2	12	6
37	3	12	6
38	1	17	0
39	3	12	6
40	1	24	0
41	1	1	0
42	1	42	0
43	2	12	1

Next steps: Generate code with X View recommended plots New interactive sheet

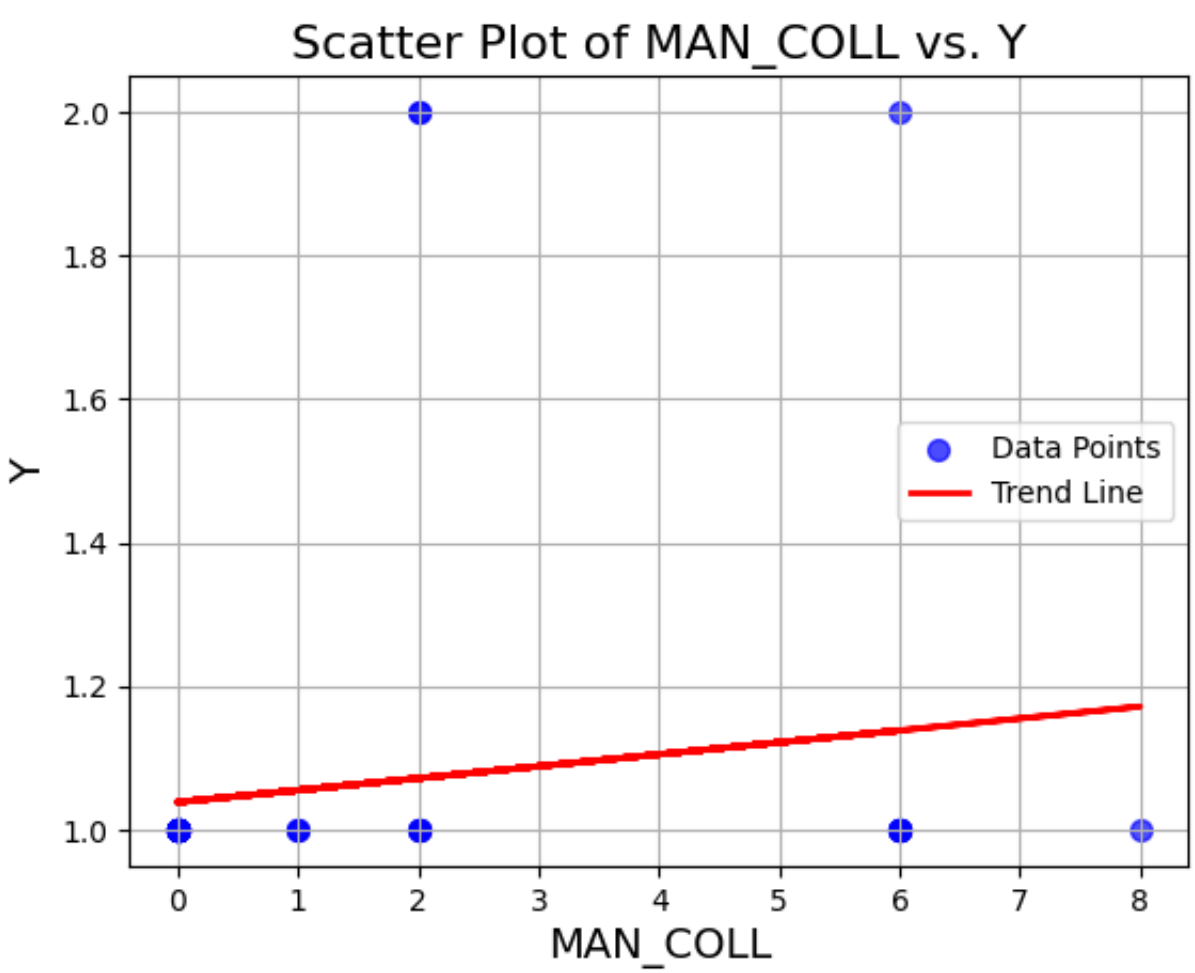
Scatterplot each combination of Independent and Dependent variables

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Create the scatter plot
plt.figure(figsize=(10, 6))  # Set figure size
plt.scatter(X['PERSONS'], Y, color='green', s=100, alpha=0.6, edgecolors='black',

# Add a title and labels
plt.title('Scatter Plot of PERSONS vs. Y', fontsize=16)
plt.xlabel('Number of Persons', fontsize=14)
plt.ylabel('Y Values', fontsize=14)

# Add gridlines for better readability
plt.grid(True)

# Optionally, add a trend line
m, b = np.polyfit(X['PERSONS'], Y, 1)  # Linear fit
plt.plot(X['PERSONS'], m * X['PERSONS'] + b, color='red', linewidth=2, label='Tre

# Add annotations for specific points (optional)
for i in range(len(X)):
    plt.annotate(f'({X["PERSONS"][i]}, {Y[i]})', (X['PERSONS'][i], Y[i]),
                 textcoords="offset points", xytext=(0,10), ha='center')

# Add a legend
plt.legend()


# Show the plot
plt.show()
```




```
import matplotlib.pyplot as plt
import numpy as np

# Sample data
# X should be a DataFrame and Y should be a Series or array
# For example:
# X = pd.DataFrame({'MAN_COLL': [1, 2, 3, 4, 5]})
# Y = np.array([5, 4, 3, 2, 1])

# Create the scatter plot
plt.scatter(X['MAN_COLL'], Y, color='blue', s=50, alpha=0.7, label='Data Points')

# Add a title and labels
```

```
plt.title('Scatter Plot of MAN_COLL vs. Y', fontsize=16)
plt.xlabel('MAN_COLL', fontsize=14)
plt.ylabel('Y', fontsize=14)

# Add gridlines for better readability
plt.grid(True)

# Optionally, add a trend line
m, b = np.polyfit(X['MAN_COLL'], Y, 1)  # Linear fit
plt.plot(X['MAN_COLL'], m * X['MAN_COLL'] + b, color='red', linewidth=2, label='T

# Add a legend
plt.legend()

# Show the plot
plt.show()
```



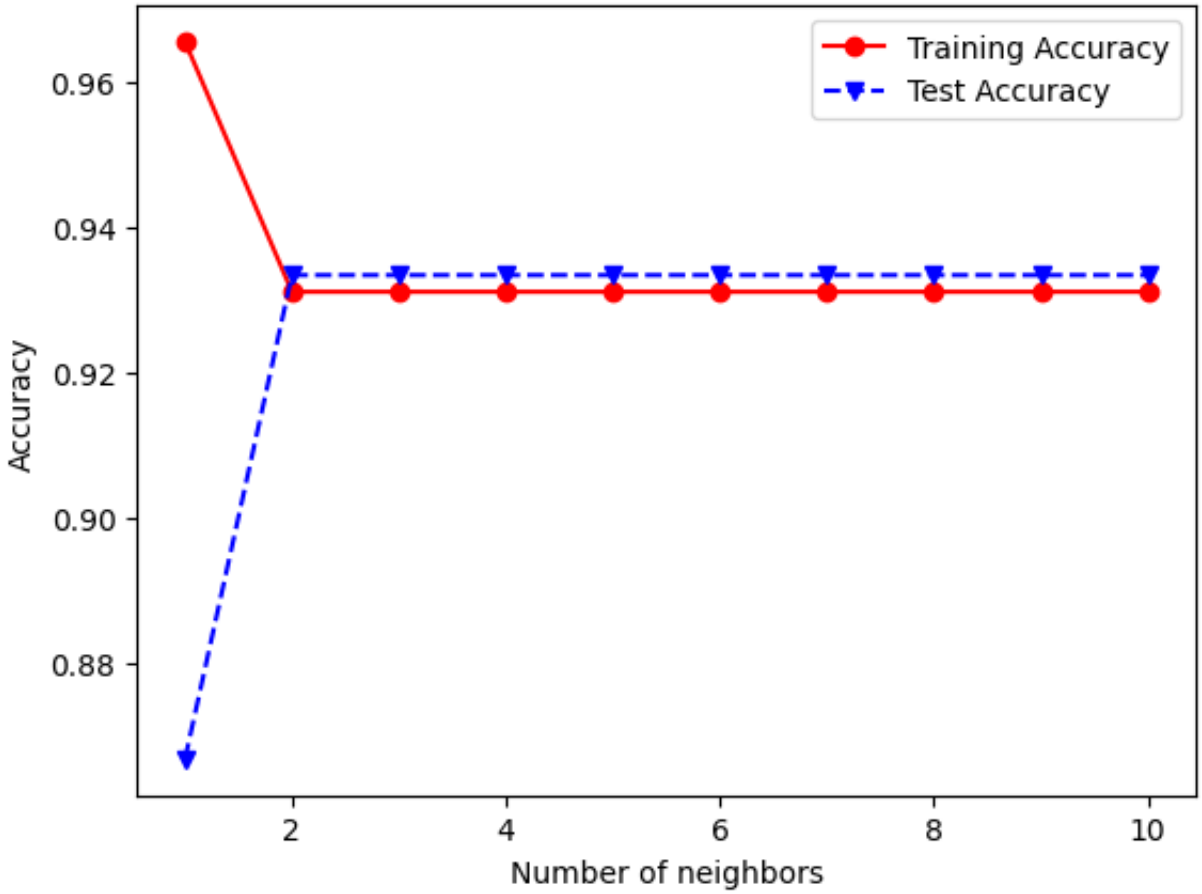
Develop your Test Train Split of the data

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_
```

Perform the Classification Analysis using 1 to 10 Nearest Neighbors.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
%matplotlib inline

numNeighbors = [1,2,3,4,5,6,7,8,9,10]
trainAcc = []
testAcc = []

for k in numNeighbors:
    clf = KNeighborsClassifier(n_neighbors=k, metric='minkowski', p=2)
    clf.fit(X_train, Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    trainAcc.append(accuracy_score(Y_train, Y_predTrain))
    testAcc.append(accuracy_score(Y_test, Y_predTest))

plt.plot(numNeighbors, trainAcc, 'ro-', numNeighbors, testAcc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
```

```
Text(0, 0.5, 'Accuracy')
```



Perform the analysis again to discover the worst columns in classifying FATALs using the same steps as above.

```
# copy and insert your code here
```

What results did your analysis Yield? Provide a brief summary below:

Double-click (or enter) to edit