HW 5

This assignment covers several aspects of Logistic Regresstion & KNN Classifier.

DO NOT ERASE MARKDOWN CELLS AND INSTRUCTIONS IN YOUR HW submission

- **Q** - QUESTION
- **A** - Where to input your answer

Instructions

Keep the following in mind for all notebooks you develop:

- Structure your notebook.
- Use headings with meaningful levels in Markdown cells, and explain the questions each piece of code is to answer or the reason it is there.
- Make sure your notebook can always be rerun from top to bottom.
- Please start working on this assignment as soon as possible. If you are a beginner in Python this might take a long time. One of the objectives of this assignment is to help you learn python and scikit-learn package.
- See README.md (README.md) for homework submission instructions

Related Tutorials

Refreshers

- Intro to Machine Learning w scikit-learn (https://scikit-learn.org/stable/tutorial/basic/tutorial.html)
- A tutorial on statistical-learning for scientific data processing (https://scikit-learn.org/stable/tutorial/statistical_inference/index.html#stat-learn-tut-index)

Classification Approaches

- Logistic Regression with Sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
- KNN with sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)
- Support Vector machine example (https://scikit-learn.org/stable/auto_examples/exercises/plot_iris_exercise.html#sphx-glr-auto-examples-exercises-plot-iris-exercise-py)
- SVC (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC)

Modeling

- Cross-validation (https://scikit-learn.org/stable/modules/cross_validation.html)

- Plot Confursion Matrix with Sklearn (https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)
- Confusion Matrix Display (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html#sklearn.metrics)

Data Processing

Q1 Get training data from the dataframe

1. Load customer_data.csv into data frame
2. Assign values of `label` column to `y`
3. Drop 'label' column from data frame,
4. Assign df values to x
5. Print the head of the dataframe

A1 Replace ??? with code in the code cell below

In [22]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
import sklearn
#Read the data/customer_data.csv file using the prropriate separator a
df = pd.read_csv('customer_data.csv')

print('The scikit-learn version is {}.'.format(sklearn.__version__))
print(df.head())


#***This was moved down to question 4 to get rid of the NaN values****
#y= df.label.values
#df.drop(columns = ['label'],inplace = True)
#x= df.values
```

```
The scikit-learn version is 1.0.2.
   label        id  fea_1   fea_2  fea_3     fea_4  fea_5  fea_6  fea
 7  \
```

In [4]:
```
#print head of x
```

Out[4]:
```
array([[5.49826650e+07, 5.00000000e+00, 1.24550000e+03, ...,
        5.00000000e+00, 1.51300000e+05, 2.44948974e+02],
       [5.90047790e+07, 4.00000000e+00, 1.27700000e+03, ...,
        3.00000000e+00, 3.41759000e+05, 2.07173840e+02],
       [5.89908620e+07, 7.00000000e+00, 1.29800000e+03, ...,
        5.00000000e+00, 7.20010000e+04, 1.00000000e+00],
       ...,
       [5.89953810e+07, 7.00000000e+00, 1.22000000e+03, ...,
        5.00000000e+00, 7.10020000e+04, 1.00000000e+00],
       [5.89980540e+07, 4.00000000e+00, 1.25000000e+03, ...,
        5.00000000e+00, 7.20000000e+04, 1.00000000e+00],
       [5.49897810e+07, 4.00000000e+00, 1.41500000e+03, ...,
        4.00000000e+00, 1.51300000e+05, 2.73861279e+02]])
```

Q2:

1. Check if there is any null value in the x dataframe.
2. If there is any column with high null values, Use a good method and replace the null values.
3. Again Check if there is any null value remaining in the dataset

`Note:` Use any of the different techniques shown in the MissingValue Handling Notebook from the last Monday Class

A2 Replace ??? with code in the code cell below

In [23]:
```
```

Out[23]:

	label	id	fea_1	fea_2	fea_3	fea_4	fea_5	fea_6	fea_7	fea_8	fea_9	fea_10	fea_
0	False	False	False	False	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	False	False	False	False	Fal
4	False	False	False	True	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	...	...	...	...	
1120	False	False	False	False	False	False	False	False	False	False	False	False	Fal
1121	False	False	False	True	False	False	False	False	False	False	False	False	Fal
1122	False	False	False	False	False	False	False	False	False	False	False	False	Fal
1123	False	False	False	False	False	False	False	False	False	False	False	False	Fal
1124	False	False	False	False	False	False	False	False	False	False	False	False	Fal

1125 rows × 13 columns

In [24]:

In [25]:

Out[25]:
```
label     0
id        0
fea_1     0
fea_2     0
fea_3     0
fea_4     0
fea_5     0
fea_6     0
fea_7     0
fea_8     0
fea_9     0
fea_10    0
fea_11    0
dtype: int64
```

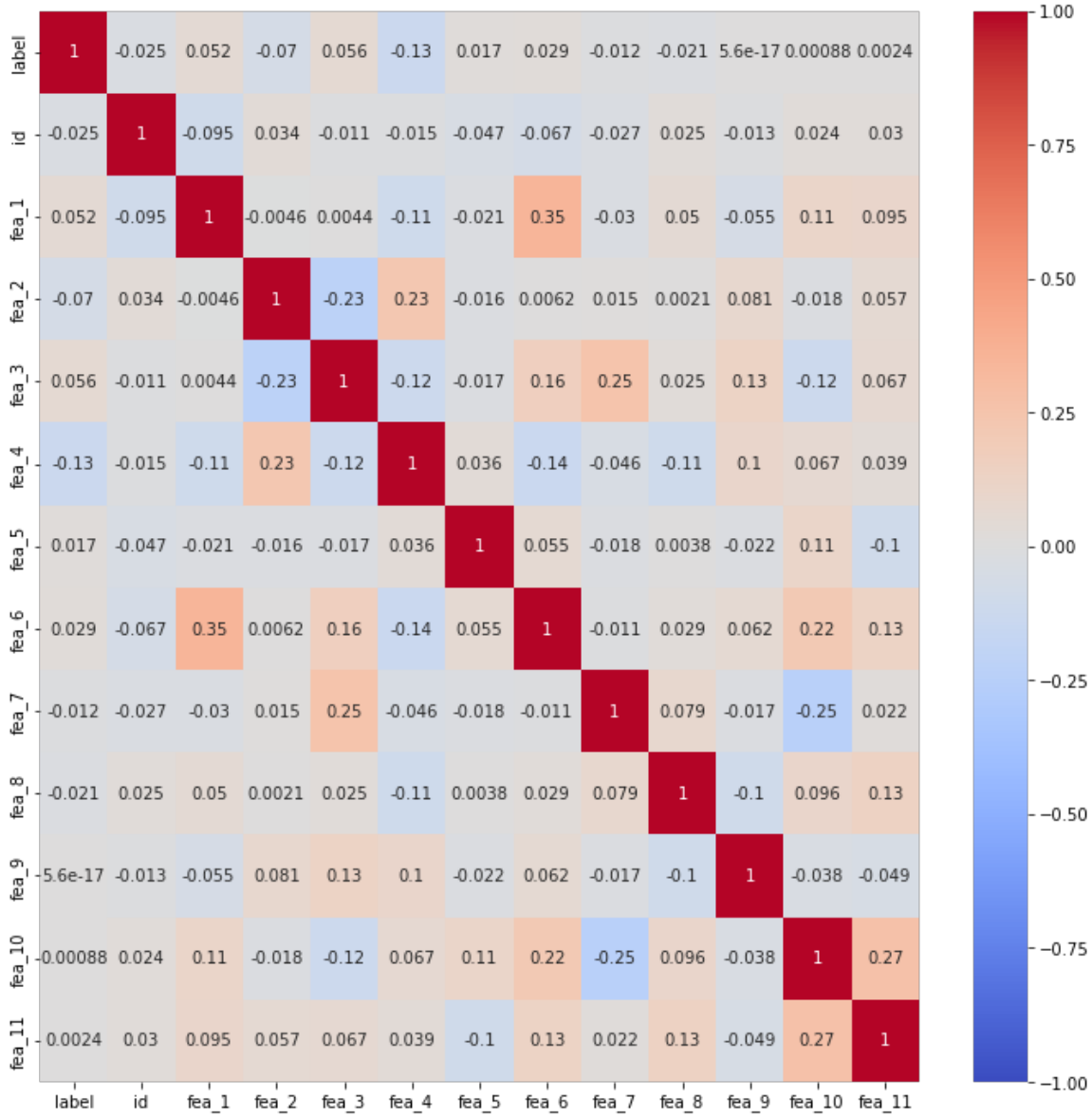
Q3: Part 1. Use heatmap chart from seaborn library to findout the correlation between the columns. Replace ??? with code in the code cell below

2. Which of the columns is mostly related to mpg column and do you think the features available from the dataset will be a good predictor?

A3 Part 1.

In [6]:
```
import seaborn as sns
plt.figure(figsize=(12,12))
```

Out[6]: <AxesSubplot:>



Q3: Part 2. Which of the columns is mostly related to mpg column and do you think the features available from the dataset will be a good predictor?

A3 Part 2 answer in this cell

Q4:

1. Split dataset into train and test data use train_test_split with test_size = 0.2 and random_state = 42
2. Check the number of instance in the train and test set.

A4 Replace ??? with code in the code cell below

In [26]:
```
#Moved in from prob. 1 due to train & test containing NaN values
y= df.label.values
df.drop(columns = ['label'],inplace = True)
x= df.values

xtrain, xtest, ytrain, ytest =  train_test_split(x,y, test_size = 0.2,
```

In [27]:
```
print(xtrain.shape)
```

```
(900, 12)
(225, 12)
```

Model 1: Logistic Regression

1. Apply Logistic Regression to our dataset
2. Show its classification accuracy using test-train splitting
3. Show its classification accuracy using K-fold cross validation

Q5 Train Logistic Regression Model

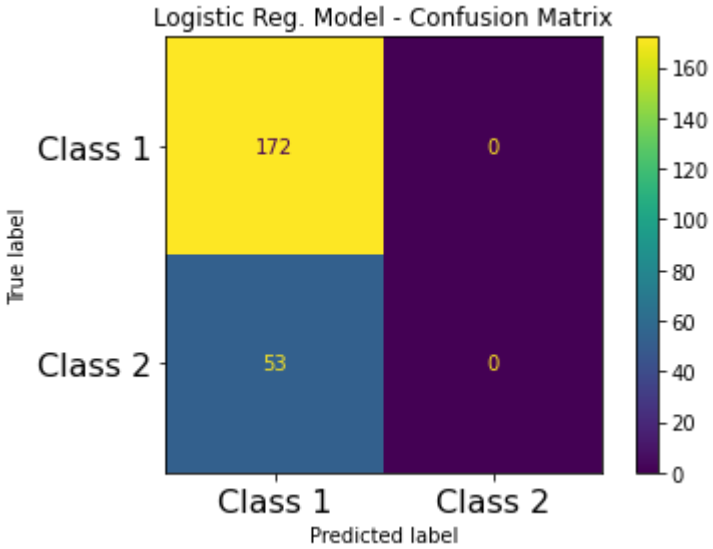
1. Create a logistic regression model using sklearn linear_model (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html) library.
2. Fit the model with the train data
3. Get the score from the model using test data
4. Plot confusion matrix using ConfusionMatrixDisplay (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html), see Visualization with Display Objects (https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_display_object_visualization.html) example.

A5 Replace ??? with code in the code cell below

In [28]:
```
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Create a logistic regression model using sklearn library
clf= LogisticRegression(random_state = 0)
clf.fit(xtrain, ytrain)



#print score for test data
```

```
0.8088888888888889
```

In [29]:
```
cm = ConfusionMatrixDisplay.from_estimator(clf, xtest, ytest)

plt.title("Logistic Reg. Model - Confusion Matrix")
plt.xticks(range(2), ["Class 1","Class 2"], fontsize=16)
plt.yticks(range(2), ["Class 1","Class 2"], fontsize=16)
```



Q6: Train Logistic Regression Model using cross-validation on `Train Data`

1. Now, use Kfold cross validation technique for the model evaluation.(`Use K=5` or try using other number of folds to see what works best)
2. Print the different scores from different folds

A6: Replace ??? with code in the code cell below

In [30]:
```
from sklearn.model_selection import cross_val_score

# Use sklearn for 5 fold cross validation
scores_log= cross_val_score(clf, xtrain, ytrain, cv = 5)

# print the scores from different folds
```

```
[0.81111111 0.81111111 0.81111111 0.80555556 0.80555556]
```

Model 2: K Nearest Neighbour Classifier

1. Apply KNN on the train dataset
2. Show its classification accuracy using test-train splitting
3. Show its classification accuracy using K-fold cross validation

Q7 Build a KNN Classification Model for the dataset

`Steps:`

1. Create a KNN model using sklearn library, and initialize n_neighbors [documentation (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html).
2. Fit the model with the train data
3. Predict the values from test data
4. Print out the score from training and test data
5. Repeat Step 1.- 4. for a range of `n_neighbors` values (k in kNN) from 1 to 40.

In [31]:

```
from sklearn.neighbors import KNeighborsClassifier


# Define KNN model
for k in range(1,40):
    knn = KNeighborsClassifier(n_neighbors = k)

    #Fit KNN model on xtrain, ytrain from above
    knn.fit(xtrain, ytrain)
    #predict y values from xtest
    y_pred=knn.predict(xtest)

    #print score for test data
    print("K: ",k,"Train Score: ",knn.score(xtrain, ytrain), "Test Scc
```

```
K:  1 Train Score:  1.0 Test Score:  0.6755555555555556
K:  2 Train Score:  0.86 Test Score:  0.7244444444444444
K:  3 Train Score:  0.8644444444444445 Test Score:  0.715555555555555
5
K:  4 Train Score:  0.8311111111111111 Test Score:  0.751111111111111
1
K:  5 Train Score:  0.8388888888888889 Test Score:  0.742222222222222
2
K:  6 Train Score:  0.8255555555555556 Test Score:  0.746666666666666
7
K:  7 Train Score:  0.8244444444444444 Test Score:  0.746666666666666
7
K:  8 Train Score:  0.8222222222222222 Test Score:  0.751111111111111
1
K:  9 Train Score:  0.8233333333333334 Test Score:  0.746666666666666
7
K:  10 Train Score:  0.8144444444444444 Test Score:  0.74222222222222
22
K:  11 Train Score:  0.8155555555555556 Test Score:  0.75111111111111
11
K:  12 Train Score:  0.8177777777777778 Test Score:  0.75111111111111
11
K:  13 Train Score:  0.82 Test Score:  0.7466666666666667
K:  14 Train Score:  0.8144444444444444 Test Score:  0.76
K:  15 Train Score:  0.8155555555555556 Test Score:  0.755555555555555
55
K:  16 Train Score:  0.81 Test Score:  0.7644444444444445
K:  17 Train Score:  0.8111111111111111 Test Score:  0.764444444444444
45
K:  18 Train Score:  0.8122222222222222 Test Score:  0.768888888888888
88
K:  19 Train Score:  0.8155555555555556 Test Score:  0.773333333333333
33
K:  20 Train Score:  0.8133333333333334 Test Score:  0.773333333333333
33
K:  21 Train Score:  0.8155555555555556 Test Score:  0.7688888888888
```

Q7 Part 2:

What is the best `n_neighbors` ? Why?

A7 The best n_neighbors from looking at the chart appears to be n = 3 as it contains teh highest training score.

Q8 Train KNN classifier using cross-validation approach, [sklearn.cross_validation (https://scikit-learn.org/stable/modules/cross_validation.html)](https://scikit-learn.org/stable/modules/cross_validation.html) tutorial.

Note:

Try a range of `n_neighbors` values (k in kNN) from 1 to 40.

A8 Replace ??? with code in the code cell below **

In [32]:

```
# Define KNN model
from sklearn.model_selection import cross_val_score
```

```
for k in range(1 , 40):
    #Define KNN model
    knn_crossval = KNeighborsClassifier(n_neighbors = k)

    # Use sklearn for 5 fold cross validation
    scores_cv= cross_val_score(knn_crossval , xtrain, ytrain, cv = 5)

    # print the scores from different folds
```

```
[0.71666667 0.75555556 0.68333333 0.71111111 0.76111111]
[0.78333333 0.8        0.79444444 0.78333333 0.8        ]
[0.71666667 0.77777778 0.78333333 0.75       0.76111111]
[0.77222222 0.79444444 0.80555556 0.80555556 0.8        ]
[0.75       0.77222222 0.79444444 0.75555556 0.79444444]
[0.79444444 0.8        0.80555556 0.79444444 0.81111111]
[0.78888889 0.79444444 0.80555556 0.78888889 0.81666667]
[0.79444444 0.8        0.80555556 0.78333333 0.81666667]
[0.79444444 0.8        0.81111111 0.78333333 0.82222222]
[0.80555556 0.80555556 0.81111111 0.79444444 0.81111111]
[0.79444444 0.80555556 0.81111111 0.79444444 0.81666667]
[0.8        0.82222222 0.81111111 0.79444444 0.81111111]
[0.80555556 0.81111111 0.81666667 0.8        0.82222222]
[0.81111111 0.81666667 0.81666667 0.79444444 0.81111111]
[0.8        0.80555556 0.81111111 0.78333333 0.81111111]
[0.80555556 0.79444444 0.81111111 0.78333333 0.81111111]
[0.80555556 0.78888889 0.81111111 0.79444444 0.81111111]
[0.78888889 0.8        0.81111111 0.79444444 0.80555556]
[0.78888889 0.78333333 0.81111111 0.78888889 0.81666667]
[0.80555556 0.81111111 0.80555556 0.8        0.80555556]
[0.81111111 0.80555556 0.81111111 0.79444444 0.81111111]
[0.81111111 0.80555556 0.81111111 0.79444444 0.80555556]
[0.81111111 0.80555556 0.81111111 0.79444444 0.81111111]
[0.81111111 0.8        0.80555556 0.79444444 0.80555556]
[0.81111111 0.80555556 0.81111111 0.79444444 0.8        ]
[0.81111111 0.81111111 0.80555556 0.79444444 0.80555556]
[0.81111111 0.81111111 0.80555556 0.79444444 0.81111111]
[0.81111111 0.81111111 0.80555556 0.79444444 0.81111111]
[0.81666667 0.81666667 0.80555556 0.79444444 0.81111111]
[0.81111111 0.81666667 0.81111111 0.8        0.81111111]
[0.80555556 0.81666667 0.81111111 0.79444444 0.81111111]
[0.81111111 0.81111111 0.81111111 0.8        0.81111111]
[0.81111111 0.81666667 0.81666667 0.79444444 0.81111111]
[0.81111111 0.81111111 0.81666667 0.81111111 0.80555556]
[0.81111111 0.81111111 0.81666667 0.8        0.80555556]
[0.81111111 0.81111111 0.81111111 0.8        0.80555556]
[0.81111111 0.81111111 0.81666667 0.79444444 0.80555556]
[0.81111111 0.81111111 0.81111111 0.79444444 0.80555556]
[0.81111111 0.81111111 0.81111111 0.79444444 0.80555556]
```

Model 3: Support Vector Machine

1. First example of how to use support vector machines for classification, link (https://scikit-learn.org/stable/modules/svm.html)

2. Classification accuracy using test-train splitting and K-fold cross validation

Q9 Create a SVM model for the train data

- fit the SVM model on the train data
- predict the values from test data
- print out the score from test data

A9 Replace ??? with code in the code cell below

In [34]:
```
from sklearn.svm import SVC

# Define SVM model
sv_svc = SVC()

#Fit SVM model on xtrain, ytrain from above
sv_svc.fit(xtrain, ytrain)

# Predict y values from xtest
pred_svc= sv_svc.predict(xtest)

#Number of instances per class
print(np.unique(pred_svc))

#print score for test data
```

```
[0]
```

Out[34]:
```
0.7644444444444445
```

Q10 Train SVM using cross-validation. See A6, A8, and [sklearn.cross_validation (https://scikit-learn.org/stable/modules/cross_validation.html)](https://scikit-learn.org/stable/modules/cross_validation.html) tutorial for hints.

A10 Replace ??? with code in the code cell below

In [33]:
```
from sklearn.model_selection import cross_val_score

# Use sklearn for 5 fold cross validation
scores_svc= cross_val_score(sv_svc, xtrain, ytrain, cv = 5)

# print the scores from different folds
```

```
[0.81111111 0.81111111 0.81111111 0.80555556 0.80555556]
```

Comparison

Q11 Compare the three models (trained using xtrain,ytrain) in terms of score.

- Train Three different models on Train data
- Predict Xtrain using the trained models
- Make a correlation matrix between Ytrain and predicted value from the Three Models

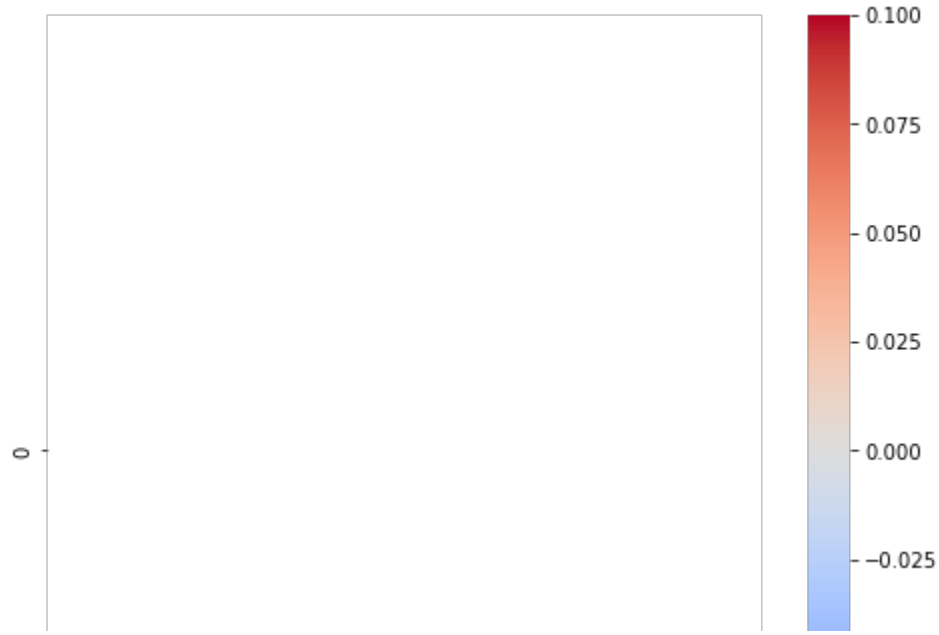
- Your resulting matrix should be `4x4 correlation matrix` for xtrain, ytrain data
  - The matrix is symmetric
  - It will provide the correlation between three model predictions plus ytrain for xtrain

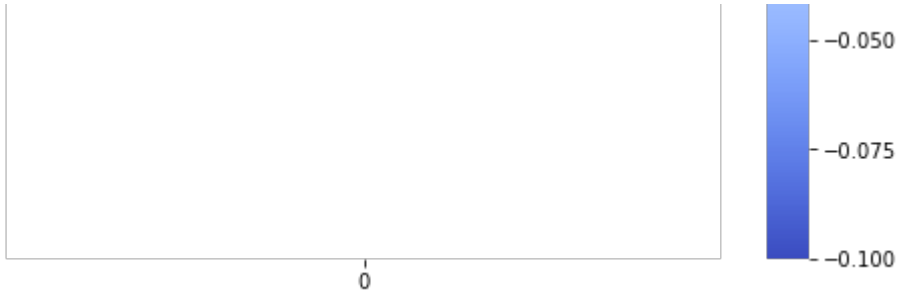
In [43]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Predict Train dataset y using logistic reg
clf= LogisticRegression()
# Fit on train data
clf.fit(xtrain, ytrain)
ypred_log= clf.predict(xtrain)

# Predict Train dataset y using KNN
knn= KNeighborsClassifier(n_neighbors=5)
knn.fit(xtrain, ytrain)
ypred_knn= knn.predict(xtrain)

# Predict Train dataset y using SVM
sv_svc=SVC()
sv_svc.fit(xtrain, ytrain)
ypred_svm= sv_svc.predict(xtrain)

labels=[0,1]

print(ytest.shape, ypred_log.shape, ypred_knn.shape, ypred_svm.shape)
# Create a dataframe using the predicted results from the models
df = pd.DataFrame({ypred_log.shape, ypred_knn.shape, ypred_svm.shape})

#copute correlation

# Now use seaborn library to plot the heatmap correlation matrix
plt.figure(figsize=(8,8))
```

```
(225,) (900,) (900,) (900,)

C:\Users\pedro\anaconda3\lib\site-packages\seaborn\matrix.py:203: Run
timeWarning: All-NaN slice encountered
  vmax = np.nanmax(calc_data)
```

Out[43]: <AxesSubplot:>




−0.050
−0.075
−0.100
0