ASSIGNMENT - 2

Question 1:

A. Load the Dataset.

```
data = 'diabetes_01_health_indicators.csv'
```

```
df = pd.read_csv(data)
print("Shape of DataFrame: ", df.shape)
df.isnull().sum()
```

No nan in any features.

```
Shape of DataFrame:  (253680, 22)

Diabetes_binary         0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
Income                  0
dtype: int64
```

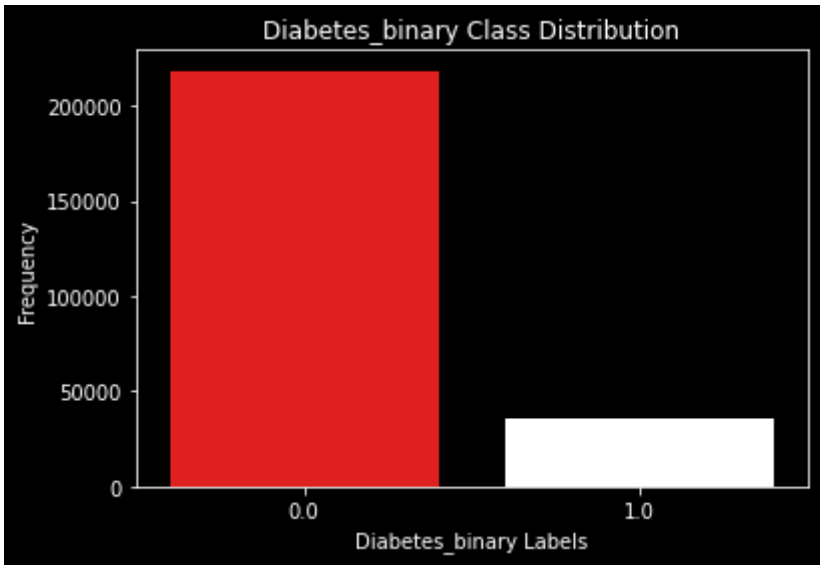
Size of dataset without Diabetes_Binary is (253680, 21)

B. Balance of the Diabetes_Binary:

```
Diabetes_binary
0.0    218334
1.0     35346
```

This is not not properly balanced, Non diabetic patients are almost 7 times the diabetic patients, which could lead imbalance of class errors.

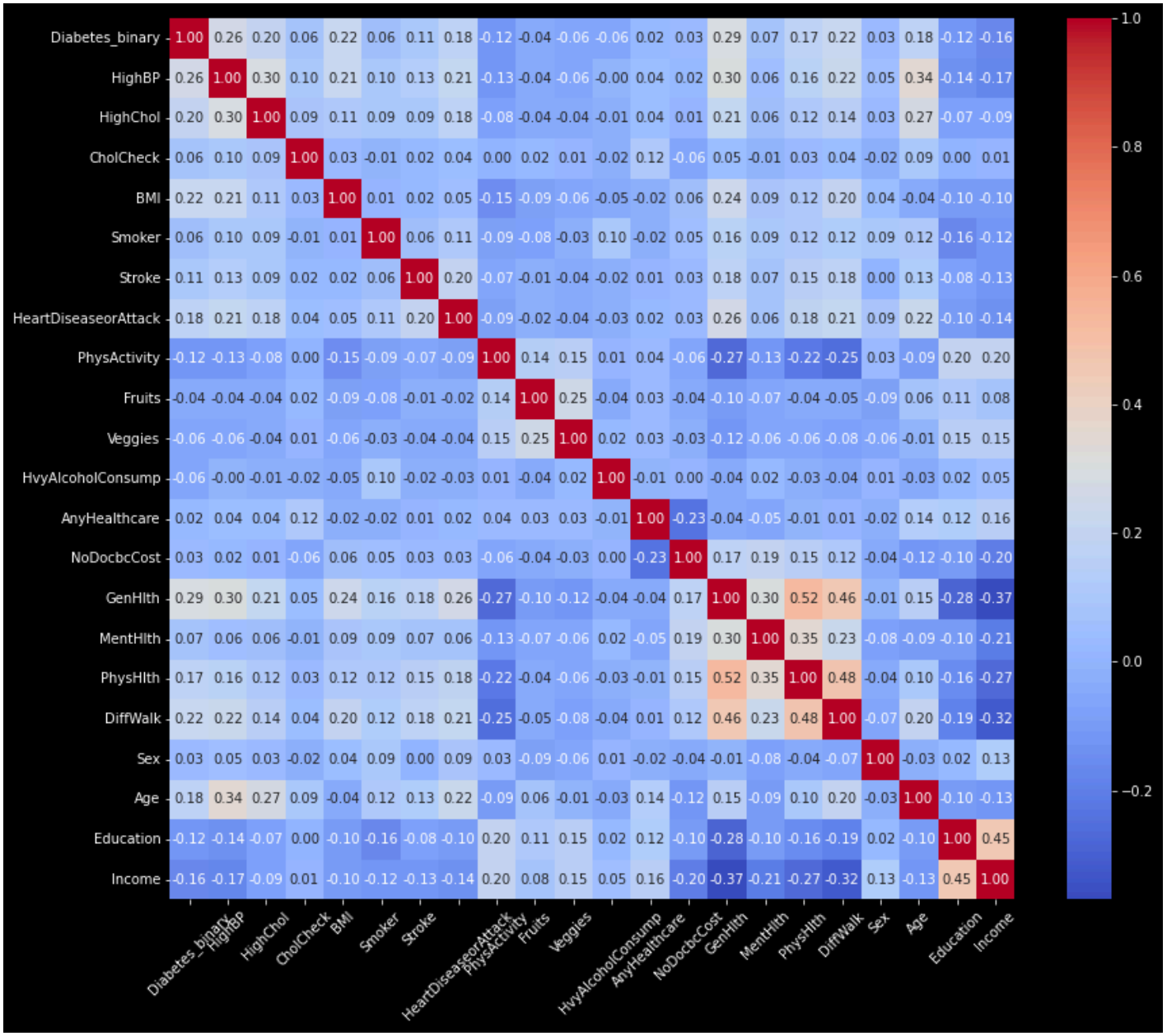
C. Visualize the class distribution



D.Descriptive Statistics:

	count	mean	std	min	25%	50%	75%	max
Diabetes_binary	253680.0	0.139333	0.346294	0.0	0.0	0.0	0.0	1.0
HighBP	253680.0	0.429001	0.494934	0.0	0.0	0.0	1.0	1.0
HighChol	253680.0	0.424121	0.494210	0.0	0.0	0.0	1.0	1.0
CholCheck	253680.0	0.962670	0.189571	0.0	1.0	1.0	1.0	1.0
BMI	253680.0	28.382364	6.608694	12.0	24.0	27.0	31.0	98.0
Smoker	253680.0	0.443169	0.496761	0.0	0.0	0.0	1.0	1.0
Stroke	253680.0	0.040571	0.197294	0.0	0.0	0.0	0.0	1.0
HeartDiseaseorAttack	253680.0	0.094186	0.292087	0.0	0.0	0.0	0.0	1.0
PhysActivity	253680.0	0.756544	0.429169	0.0	1.0	1.0	1.0	1.0
Fruits	253680.0	0.634256	0.481639	0.0	0.0	1.0	1.0	1.0
Veggies	253680.0	0.811420	0.391175	0.0	1.0	1.0	1.0	1.0
HvyAlcoholConsump	253680.0	0.056197	0.230302	0.0	0.0	0.0	0.0	1.0
AnyHealthcare	253680.0	0.951053	0.215759	0.0	1.0	1.0	1.0	1.0
NoDocbcCost	253680.0	0.084177	0.277654	0.0	0.0	0.0	0.0	1.0
GenHlth	253680.0	2.511392	1.068477	1.0	2.0	2.0	3.0	5.0
MentHlth	253680.0	3.184772	7.412847	0.0	0.0	0.0	2.0	30.0
PhysHlth	253680.0	4.242081	8.717951	0.0	0.0	0.0	3.0	30.0
DiffWalk	253680.0	0.168224	0.374066	0.0	0.0	0.0	0.0	1.0
Sex	253680.0	0.440342	0.496429	0.0	0.0	0.0	1.0	1.0
Age	253680.0	8.032119	3.054220	1.0	6.0	8.0	10.0	13.0
Education	253680.0	5.050434	0.985774	1.0	4.0	5.0	6.0	6.0
Income	253680.0	6.053875	2.071148	1.0	5.0	7.0	8.0	8.0

E.Calculate the correlation matrix and generate heatmaps to visualize correlations. Comment on any multicollinearity, if exists.



Any corelation that has a value graeter than 0.7 or less than -0.7, tells us about the existence of collinearity. A little moderate corelation is observed between PhysActivity and GenHealth. In the plot heatmap can lead us, if there is any multicolinearity by looking at the colors of the entries.

Question 2:

A.LogisticRegression No penality

```
LRnone = LogisticRegression(penalty=None, solver='lbfgs', max_iter=1000)
```

We used 'lbfgs' solver which means the loss function.

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.98      0.93     43667
         1.0       0.54      0.16      0.24      7069

    accuracy                           0.86     50736
   macro avg       0.71      0.57      0.59     50736
weighted avg       0.83      0.86      0.83     50736

Confusion Matrix:
 [[42717   950]
 [ 5950  1119]]
```

This model achecied a accuracy of 86%, while the dataset is imbalance with majority samples being non diabetic.

This performs well in figuring out non diabetic patients but could struggle with finding diabetic. This model is absolutely leaning in one direction which is non diabetic, it failed 84% of times which can be figured out from recall.

B. Logistic Regression with L1 penalty.

```
Accuracy with L1 regularization: 0.8639
Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.98      0.93     43667
         1.0       0.54      0.16      0.24      7069

    accuracy                           0.86     50736
   macro avg       0.71      0.57      0.58     50736
weighted avg       0.83      0.86      0.83     50736

Confusion Matrix:
 [[42717   950]
 [ 5953  1116]]
```

The accuracy is almost similar to no penalty, the lower f1 and recall of diabetic class, indicate that this model has not been any different or improved from no penalty model.

C.Logistic Regression with L2 penalty:

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.98      0.93     43667
         1.0       0.54      0.16      0.24      7069

    accuracy                           0.86     50736
   macro avg       0.71      0.57      0.58     50736
weighted avg       0.83      0.86      0.83     50736

Confusion Matrix:
 [[42715   952]
 [ 5952  1117]]
```

This model also suffered the same as the previous models, greate accuracy but very bad F1 and recall of diabetic class. None of the models have shown any improvement alos performed pretty similar.

D. impact of regularization on model performance

All models have performed pretty similarly with achieving high accuracy which is due to heavy unbiased nature or class imbalance nature of the dataset with fulcrum shifting completely towards the non-diabetic patients side.

Every model had very good F1 and recall for non diabetic but has utterly low values for diabetic class, the non diabetic has better results or better chances of being predicted because the non diabetic patients are around 85%. None of these regularisation techniques have helped in any sense, techniques like SMOTE which is a resampling technique would have helped the case.

Question 3:
A.Implement SVM with a linear kernel.
We used LineaeKernal from svm.LinearSVC()

```
from sklearn import svm
Linear = svm.LinearSVC()
```

```
# Train the model
Linear.fit(X_train, Y_train)
```

```
LinearSVC
LinearSVC()
```

```
Accuracy of Linear SVM LINEAR: 0.8633
Classification Report LINEAR:
              precision    recall  f1-score   support

         0.0       0.87      0.99      0.93     43667
         1.0       0.59      0.06      0.12      7069

    accuracy                           0.86     50736
   macro avg       0.73      0.53      0.52     50736
weighted avg       0.83      0.86      0.81     50736

Confusion Matrix LINEAR:
 [[43344   323]
 [ 6613   456]]
```

B.Implement SVM with a polynomial kernel.

```
poly = svm.SVC(kernel='poly', degree=3, C=1.0)
poly.fit(X_train, Y_train)
```

```
Accuracy of Linear SVM POLY: 0.8649
Classification Report of POLY:
              precision    recall  f1-score   support

         0.0       0.87      0.99      0.93     20675
         1.0       0.64      0.06      0.10      3325

    accuracy                           0.86     24000
   macro avg       0.75      0.53      0.52     24000
weighted avg       0.84      0.86      0.81     24000

Confusion Matrix POLY:
 [[20568   107]
 [ 3135   190]]
```

C. .Implement SVM with RBF kernel.

```
rbf = svm.SVC(kernel='rbf', C=1.0, gamma='scale')
rbf.fit(X_train, Y_train)
```

```
Accuracy of Linear SVM RBF: 0.8620
Classification Report of RBF:
              precision    recall  f1-score   support

         0.0       0.87      0.99      0.93      1728
         1.0       0.45      0.06      0.11       272

    accuracy                           0.86      2000
   macro avg       0.66      0.53      0.52      2000
weighted avg       0.81      0.86      0.81      2000

Confusion Matrix RBF:
 [[1707   21]
 [ 255   17]]
```

D. Compare the performance of each kernel using metrics such as accuracy, precision, recall, F1- score and confusion matrix. Include any visualizations if applicable. Summarize which SVM performed best and why??

All the models showed very good Accuracy but same like logistic regression models failed in predicting diabetic classes, over all the SVM models, this is majorly because of heavy class imbalance in the dataset.

F1 and recall of non-diabetic cases is excellent while too bad in Diabetic.

Sampling is done here because the complexity of polynomial SVM and RBF SVM is O(n^2), where around 2 lakh samples, which is very expensive, so the sampling is done with proper stratification.

Question 4:

A. Design and test two different neural network architectures, one being a simple feedforward neural network (e.g., 1-2 hidden layers) and the other being a deeper neural network (e.g., 3+ hidden layers, dropout for regularization)

Simple Neural Network

The simple neural network consists of one hidden layer and 1 neurons, and sigmoid activation function is used here. In both hidden and output layers, sigmosid function is used for predicting the output.

```
class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(D_tr.shape[1], 16),
            nn.Sigmoid(),
            nn.Linear(16, 1),
            nn.Sigmoid()
        )
    def forward(self, x):
        return self.model(x)
```

We used 5000 epochs with 0.01 Learning rate and Adam optimiser is used here.

```
Training Simple Neural Network...
Epoch 1/5000 - Loss: 0.70085013
Epoch 2/5000 - Loss: 0.68936050
Epoch 501/5000 - Loss: 0.40199226
Epoch 1001/5000 - Loss: 0.34162593
Epoch 1501/5000 - Loss: 0.32627851
Epoch 2001/5000 - Loss: 0.31994107
Epoch 2501/5000 - Loss: 0.31622767
Epoch 3001/5000 - Loss: 0.31357539
Epoch 3501/5000 - Loss: 0.30541039
Epoch 4001/5000 - Loss: 0.30026719
Epoch 4501/5000 - Loss: 0.29686347
Final Accuracy: 0.8555
```

We can see that as the epochs increase loss is being decreased. The loss function used is Binary cross-entropy loss.

```
Simple NN Accuracy: 0.8555
Simple NN Classification Report:
              precision    recall  f1-score   support

         0.0       0.83      0.90      0.86     43667
         1.0       0.89      0.81      0.85     43667

    accuracy                           0.86     87334
   macro avg       0.86      0.86      0.86     87334
weighted avg       0.86      0.86      0.86     87334

Simple NN Confusion Matrix:
 [[39207  4460]
 [ 8161 35506]]
```

Has an accuracy of around 85% and also significantly performed better than the previous models we discussed, which can be observed in F1, recall of both non diabetic and diabetic, infactly performed similary in predicting diabetic and non diabtic, if you observe the training the loss is also getting down, if you could properly play with number of neuron, Learning rate, and epochs, this could be potential winner among the other methods.

Deep neural network:
We used 4 hidden layers in this model, with most layers having 32 neurons and 16 neurons. We couldn't experiment more because of the computational dependencies, but here we used sigmod, Relu activation function along with dropouts.

1st hidden layer = 32 neurons with sigmoid Activation and a dropout of 30% neurons
2nd hidden layer = 32 neurons with sigmoid Activation and a dropout of 20% neurons
3rd hidden layer = 16 neurons with sigmoid Activation.
4th hidden layer = 16 neurons with Relu Activation.
OUTPUT LAYER = one neuron with sigmoid function for binary classification.

Dropout are used to avoid overfitting.

```
Training Deep Neural Network...
Epoch 1/5000 - Loss: 0.69311523
Epoch 2/5000 - Loss: 0.69276953
Epoch 501/5000 - Loss: 0.46347407
Epoch 1001/5000 - Loss: 0.42134956
Epoch 1501/5000 - Loss: 0.40705109
Epoch 2001/5000 - Loss: 0.39402795
Epoch 2501/5000 - Loss: 0.39039010
Epoch 3001/5000 - Loss: 0.38784999
Epoch 3501/5000 - Loss: 0.38758001
Epoch 4001/5000 - Loss: 0.38472173
Epoch 4501/5000 - Loss: 0.38270065
Final Accuracy: 0.5075
```

```
Deep NN Accuracy: 0.5075
Deep NN Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.02      0.03     43667
         1.0       0.50      1.00      0.67     43667

    accuracy                           0.51     87334
   macro avg       0.69      0.51      0.35     87334
```

We used 5000 epochs with 0.01 Learning rate and Adam optimiser is used here and the loss function used is Binary cross-entropy loss.

B. Compare the performance of these architectures and provide a summary of the architectures, training process, and evaluation metrics (accuracy, loss, precision, recall, F1-score, confusion matrix etc.)

Here the accuracy is around 50%, so the model is not performing any better than simple neural network. Also, for the majority class which is non-diabetic, the recall is 0.02 and the f1 is 0.03 which is the worst among all the models we have worked so far. While the model is performing supremely good in predicting diabetic casses, this demosntrates the model is blinded in predicting the nondiabetic cases.
This could be because of overfitting arised from diabetic samples, this can be cleared out with proper tuning of hyperparameters.

Question 5.
A.Implement k-fold cross-validation on Logistic regression and SVM models with different values of k (e.g., 5, 10) and provide your observations.

```python
results = []

for k in [5, 10]:
    print("********")
    print(f"**Cross-validation with k={k}:**")

    cv = StratifiedKFold(n_splits=k, shuffle=True, random_state=42)
    scLR_strat = cross_val_score(LRnone, D, L, cv=cv, scoring='accuracy').mean()
    scSVM_strat = cross_val_score(Linear, D, L, cv=cv, scoring='accuracy').mean()

    kf = KFold(n_splits=k, shuffle=True, random_state=42)
    scLR_non = cross_val_score(LRnone, D, L, cv=kf, scoring='accuracy').mean()
    scSVM_non = cross_val_score(Linear, D, L, cv=kf, scoring='accuracy').mean()

    # Add to results
    results.append(['Logistic Regression', k, scLR_strat, scLR_non])
    results.append(['SVM (Linear Kernel)', k, scSVM_strat, scSVM_non])


df_results = pd.DataFrame(results, columns=['Model', 'k', 'Stratified Accuracy', 'Not-Stratified Accuracy'])
print("\nCross-Validation Accuracy Results:\n")
print(df_results.to_string(index=False))
```

```
Cross-Validation Accuracy Results:

                Model  k  Stratified Accuracy  Non-Stratified Accuracy
Logistic Regression  5             0.863517                 0.863521
SVM (Linear Kernel)  5             0.863446                 0.863454
Logistic Regression 10             0.863509                 0.863580
SVM (Linear Kernel) 10             0.863434                 0.863434
```

We used 5-fold cross-validation, 10-fold cv and stratified and not stratified for the scenarios. Although the dataset is very imbalance, but nonstratified is observed to have a little accuracy more than stratified counterparts, has an edge by very little margin.

Theoretically stratified should perform better because this is heavy imbalance dataset, sometime you can have cases where you have zero diabetic cases, so stratification is important , especially in such heavy imbalance datasets.

RUN ques1.ipnyb fro QUESTION 1.
RUN ques2.ipnyb fro QUESTION 2.
RUN ques3.ipnyb fro QUESTION 3.
RUN ques4.ipnyb fro QUESTION 4.
RUN ques5.ipnyb fro QUESTION 5.