Credit Task 1 (Task C1),

SIG 720 – Machine Learning

On Track Submission

Submitted by:

Student Name: Surya Pradeep Kumar

Deakin Id: 223020011

Attempt # 1

Date: 23 July '23

Target Grade: C (Credit)

Index

Task Details:

Credit Task 1 - Analysis and modeling on two datasets (SCADI and Obesity Levels)

Part I

i. Load the Data:

```
df = pd.read_csv("./SCADI.csv")
df.head(7)
```

	Gender	Age	d 5100-0	d 5100-1	d 5100-2	d 5100-3	d 5100-4	d 5100-8	d 5100-9	d 5101-0	...	d 57022-8	d 57022-9	d 571-0	d 571-1	d 571-2	d 571-3	d 571-4	d 571-8	d 571-9	Classes
0	0	18	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	1	0	0	class6
1	0	22	0	0	0	0	1	0	0	0	...	0	0	0	0	0	1	0	0	0	class6
2	0	18	0	0	0	1	0	0	0	0	...	0	0	0	0	0	1	0	0	0	class6
3	1	18	0	0	0	0	1	0	0	0	...	0	0	0	0	1	0	0	0	0	class6
4	0	19	0	0	0	0	1	0	0	0	...	0	0	0	0	1	0	0	0	0	class6
5	0	18	0	1	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	class2
6	0	15	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	1	0	0	class6

7 rows × 206 columns

We have a large number of dimensions w.r.t data points, just 70 rows and 205 different features and one target class. This seems to be the problem of the curse of dimensionality. We have to address this by performing dimensionality reduction later.

```
y = df["Classes"]
df["Classes"].value_counts()
```

```
Classes
class6    29
class7    16
class4    12
class2     7
class5     3
class1     2
class3     1
Name: count, dtype: int64
```

Classes are imbalanced; There are 7 different groups with a imbalanced dataset

1. Subgroups according to columns 3 to 205

```
X_subset = X.drop(["Gender", "Age"], axis=1)
X_subset.head(3)  # 203 columns from 3 to 205
```

	d 5100-0	d 5100-1	d 5100-2	d 5100-3	d 5100-4	d 5100-8	d 5100-9	d 5101-0
0	0	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0
2	0	0	0	1	0	0	0	0

3 rows × 203 columns

Subgroups using no. of unique values in selected columns

```
num_subgroups = X_subset.nunique().sum()

num_subgroups
```

```
343
```

Subgroups using no. of unique values in selected columns -- We can see that there are **343** different subgroups when considering 3 to 205 columns. And we have seen that from the 206th attribute. However, We only have **7 unique classes in the labels.**

Also, trying clustering using k-means with kmeans++ initialization method, We find that we can divide the data into **6 clusters** optimally. This was done through silhouette analysis as we have relatively good avg. silhouette coefficient and purity score along with consistent groups in the silhouette plot. ***This also does not exactly match with the number of unique classes in the 206th column i.e, the ground truth.***

This is because the separation of some of the classes in the gt might not be too good because we have too few data points in a few classes. Also, overall there are very few data points to be able to infer a meaningful pattern out of them. Along with a curse of dimensionality problem.

2. Curse of Dimensionality problem

The term "**Curse of Dimensionality**" refers to the explosive nature of increasing data dimensions and the subsequent exponential increase in computer work required for processing and/or analysis. Richard E. Bellman coined the term to describe the increase in volume of Euclidean space associated with adding extra dimensions in the field of dynamic programming. This phenomena is now being observed in domains such as machine learning, data analysis, and data mining, to mention a few. In principle, increasing the dimensions adds more information to the data, boosting its quality, but it actually increases noise and redundancy during analysis.

A feature of an item in machine learning might be an attribute or a characteristic that defines it. Each characteristic represents a dimension, and a collection of dimensions forms a data point. This is a feature vector that defines the data point that will be used by a machine learning algorithm (or algorithms). When we talk about increasing dimensionality, we mean increasing the amount of characteristics utilized to describe the data. In the realm of breast cancer research, for example, age and the number of malignant nodes can be used as features to determine a patient's prognosis. A feature vector's dimensions are made up of these features. However, other factors such as previous surgeries, patient history, tumor kind, and other such characteristics assist a doctor in making a diagnosis are adding dimensions to data.

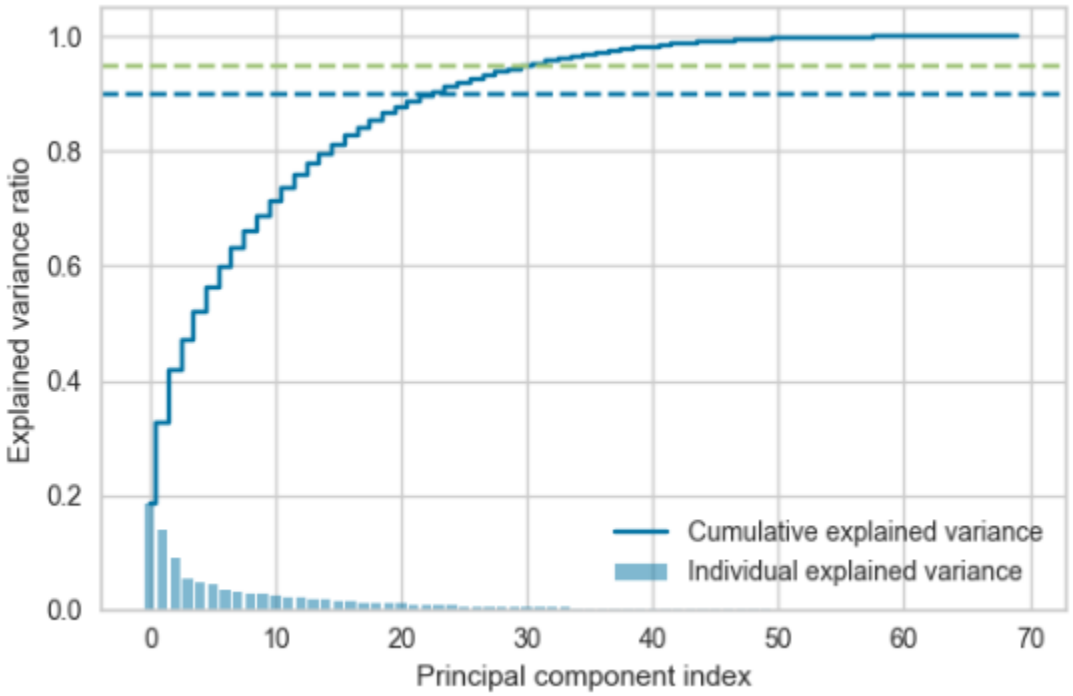
Hughes (1968) in his study concluded that with a fixed number of training samples, **the predictive power of any classifier first increases as the number of dimensions increases, but after a certain value of number of dimensions, the performance deteriorates**. Thus, the phenomenon of curse of dimensionality is also known as Hughes phenomenon.

A range of approaches known as '**Dimensionality reduction techniques**' are employed to alleviate the **issues associated with high dimensional data**.

Dimensionality reduction approaches are classified into two types: "feature selection" and "feature extraction."

- **Feature Selection**: Low Variance Filter, High Correlation Filter, Multicollinearity, Feature Ranking
- **Feature Ranking**: PCA, Factor Analysis, Independent component analysis, t-SNE

- **PCA**

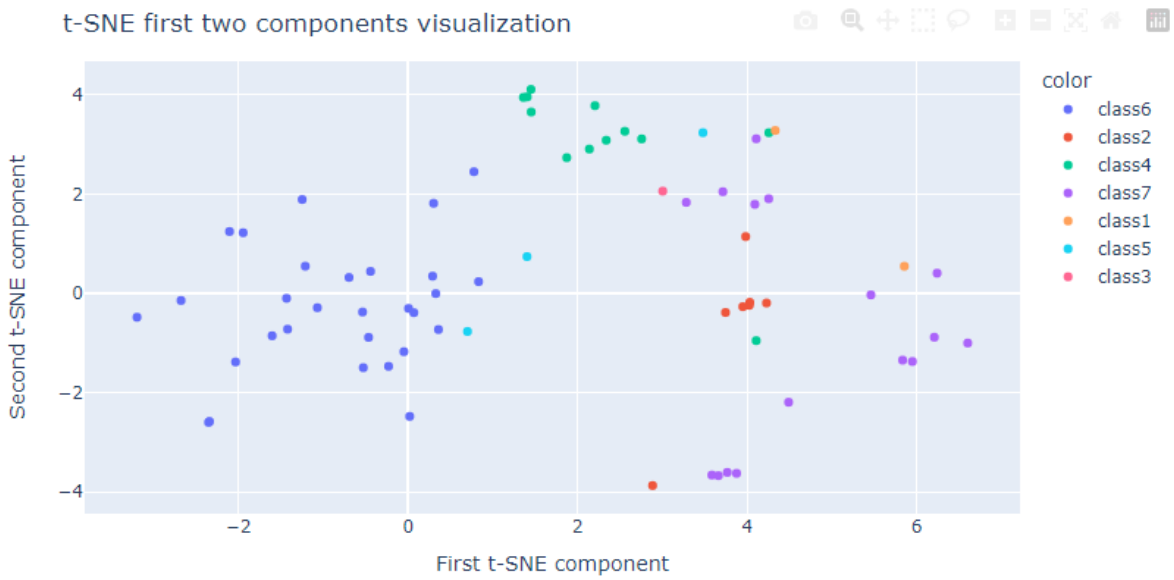


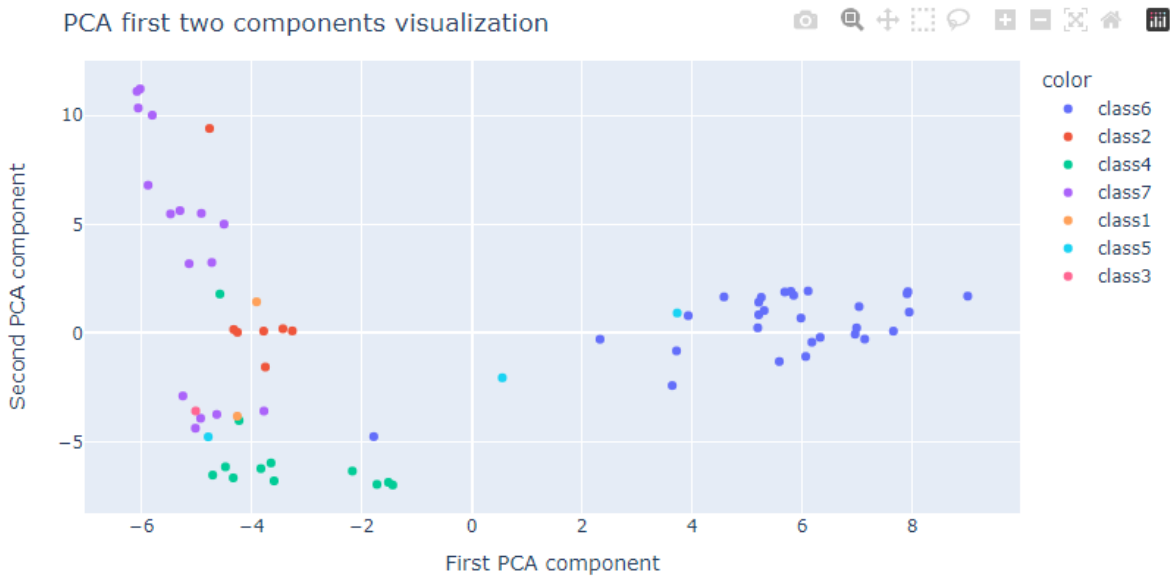
This plot shows the cumulative percentage of variance explained by each additional principal component. We can see that around 15 principal components are enough to account for approx. 82% of the variance. Also, around 23 principal components explain 90% of the variance and 30 principal components are able to explain a total of 95% of variance of the data. This means, after doing dimensionality reduction with PCA we're able to effectively use just 30 principal components instead of the original 70 dimensions.

To illustrate the problem, let's consider a two-dimensional plot. Since the dataset has 203 dimensions, we cannot directly visualize it in a traditional scatter plot.
However, we can apply dimensionality reduction techniques like PCA and t-SNE to project the dataset onto a lower-dimensional space, such as a 2D space. By applying PCA and t-SNE, we can transform the dataset into a reduced number of components while preserving the most important information. Then, We can then plot the data in this reduced 2D space.

We can see that the clusters are indeed clearly separated when visualized using just two components of PCA and t-SNE. Thus confirming the presence of the curse of dimensionality problem.

The loss of information can be measured by comparing the explained variance ratio of the original data with that of the reduced data (in 2D). The reduction in the explained variance indicates the loss of information due to dimensionality reduction
It's important to note that such a plot will only capture a subset of the information present in the original high-dimensional dataset, resulting in a loss of information.





3. Computation of Variance Explained by the Principal Components

The percentage of variance for the first N components in PCA is computed based on the eigenvalues of the covariance matrix. Steps:

- Compute the covariance matrix of the original dataset
- Perform an eigendecomposition (SVD) of the covariance matrix, which yields eigenvalues and eigenvectors.
- Sort the eigenvalues in descending order and calculate the total sum of all eigenvalues.
- Compute the cumulative sum of the eigenvalues up to the Nth component.
- After getting the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. Percentage of Variance: The percentage of variance explained by each principal component is calculated by dividing its eigenvalue by the sum o **The percentage of variance explained by the first N components indicates how much information is retained when reducing the dataset dimensionality.**f all eigenvalues.

Percentage of Variance Explained by PCi = (Eigenvalue of PCi) / (Sum of all Eigenvalues)

Cumulative Variance: Often, we are interested in the cumulative variance explained by a subset of principal components. This is useful to determine how much information is retained when using a certain number of principal components. The cumulative variance for the first k principal components is obtained by summing the percentage of variance explained by those components.

Cumulative Variance (N) (X%) = Sum of the Percentage of VariaNce for the first N principal components

By analyzing the percentage of variance explained by each principal component, one can **make an informed decision about how many principal components to retain for dimensionality reduction. Higher percentages indicate that a larger proportion of the original dataset's variance is captured by the reduced components after reduction**. Typically, a cumulative variance of around 95% or higher is considered a good choice, as it retains most of the information from the original data while reducing the dimensions.

Part II

4. ML Modelling

```
df = pd.read_csv("ObesityDataSet.csv")
df.shape
```

```
(2111, 17)
```

```
df.info()  # no missing values in dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Gender                          2111 non-null   object
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object
 5   FAVC                            2111 non-null   object
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object
 9   SMOKE                           2111 non-null   object
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            2111 non-null   object
 15  MTRANS                          2111 non-null   object
 16  NObeyesdad                      2111 non-null   object
```

Dataset description: This dataset includes data for the estimation of obesity levels in individuals based on their eating habits and physical condition. The data contains 17 attributes and 2111 records.

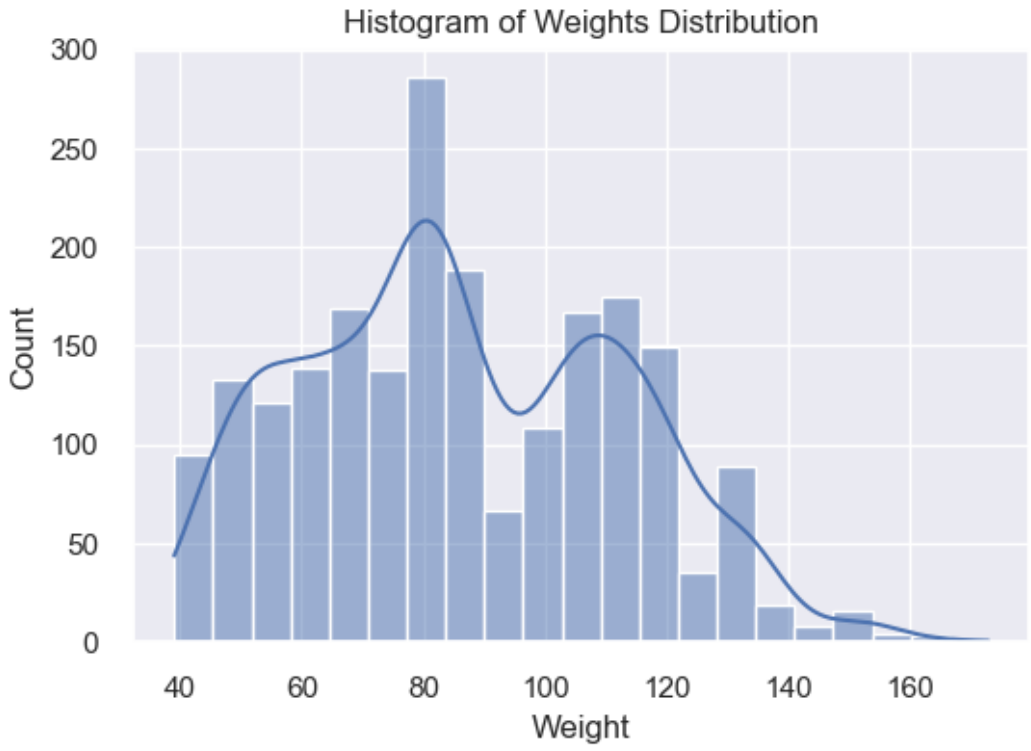
```
# dropping NObeyesdad feature as required

df.drop(["NObeyesdad"], inplace=True, axis=1)
df.shape
```

```
(2111, 16)
```

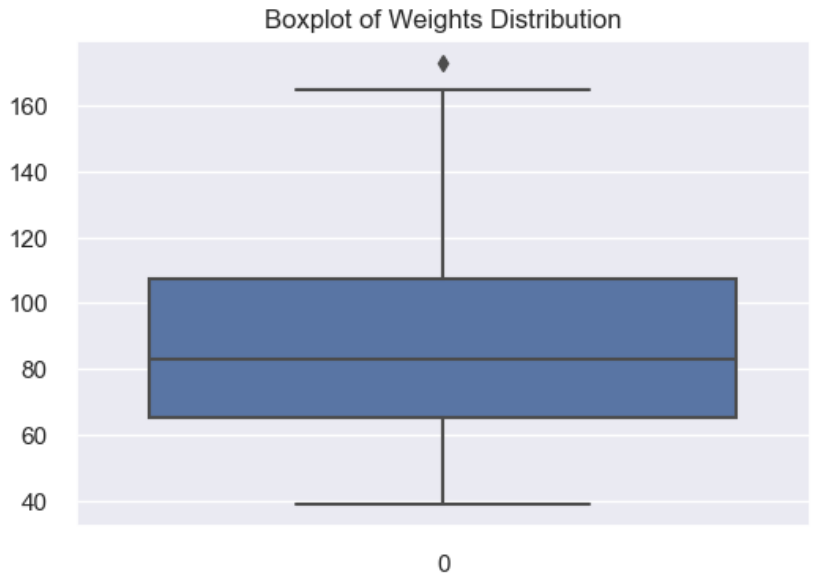
```
# target variable for prediction is "weight"
# seperating features and target

y = df["Weight"]
X = df.drop(["Weight"], axis=1)
X.shape, y.shape
```

```
((2111, 15), (2111,))
```

Distribution of Target:



It seems to have a bimodal normal distribution for the weight. There do not seem to be many outliers present as well.



From the boxplot, there are not many outliers as well

i. Data Wrangling

From the pair plot: Age, Height, Weight seem to be normally distributed. We can't directly infer heavy correlation between the scatter plots except between Height and Weight which is expected.

```
X.select_dtypes(include=["object"])  # categorical columns
```

	Gender	family_history_with_overweight	FAVC	CAEC	SMOKE	SCC	CALC	MTRANS
0	Female	yes	no	Sometimes	no	no	no	Public_Transportation
1	Female	yes	no	Sometimes	yes	yes	Sometimes	Public_Transportation
2	Male	yes	no	Sometimes	no	no	Frequently	Public_Transportation
3	Male	no	no	Sometimes	no	no	Frequently	Walking
4	Male	no	no	Sometimes	no	no	Sometimes	Public_Transportation
...	...	...	...	...	...	...	...	...
2106	Female	yes	yes	Sometimes	no	no	Sometimes	Public_Transportation
2107	Female	yes	yes	Sometimes	no	no	Sometimes	Public_Transportation
2108	Female	yes	yes	Sometimes	no	no	Sometimes	Public_Transportation
2109	Female	yes	yes	Sometimes	no	no	Sometimes	Public_Transportation
2110	Female	yes	yes	Sometimes	no	no	Sometimes	Public_Transportation

These all seem to be textual, but they are actually just categorical columns, so we can encode them by one-hot encoding or label encoder. There is also an inherent imbalance in all other columns excluding gender.

```
X_le.isna().sum()  # no missing values, no need to impute or drop
```

```
Gender                            0
Age                               0
Height                            0
family_history_with_overweight    0
FAVC                              0
FCVC                              0
NCP                               0
CAEC                              0
SMOKE                             0
CH2O                              0
SCC                               0
FAF                               0
TUE                               0
CALC                              0
MTRANS                            0
dtype: int64
```

There are no missing values to handle as well.

ii. Modeling

Modeling was done by using linear regression, support vector regression and also random forest regression. Of these, random forest performed best after finding the best hyper parameters using randomized search. Also, the best model was trained using cross-validation instead of a simple split to obtain an **mse error of 64 and R2 score of 0.9.** Other metrics for the models are reported in the notebook.

a. Model Selection:

The model selected as the champion for prediction of weight in the obesity levels dataset is the Random Forest Regressor model. Random Forest is an ensemble learning method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It is suitable for regression tasks and works well with a mix of numerical and categorical features. It's also a good model which is robust against overfitting. Also, because it's an ensemble model, the variance would also be low.

b. Assumption for the Target Variable:

The distribution of target variable "weight" was checked and it's found to be a float with a normal distribution, as it represents the weight of individuals. Regression models are appropriate for predicting continuous target variables. Hence, we conclude we need to fit a regression model to predict the target.

c. Handling Text Variables:

Raw text or natural text variables aren't present in this dataset. Those textual features need to be preprocessed and transformed into numerical features before feeding them to the ML model. Generally, bag of words, tf-idf, word embeddings, or using pre-trained language models like Word2Vec, BERT or flair is preferred. But in the obesity dataset, we only have categorical columns. THen, we can do label encoding to just convert into integers. But, if the categorical features had an inherent order as well, we should've used one hot encoding for those variables.

d. Model Parameter Optimization:

In this code, we have optimized the model hyper parameters for all the model experiments using Grid Search or Randomized Search as appropriate. For the Random Forest Regressor, we have found the best combination of hyperparameters, such as the number of estimators (trees), max depth, min samples split, min leaves etc., that yields the best performance.

e. Handling Overfitting or Underfitting:

We have seen underfitting in linear regression and support vector regression models. As they didn't have enough complexity to handle the dataset. Additionally, feature engineering and selection can be done to include more relevant features in the model to prevent underfitting.

Random forest regressor is naturally robust against overfitting. Increasing the minimum samples required to split a node, limiting the maximum depth of the tree, or increasing the number of estimators can help prevent overfitting if present.

It is important to evaluate the model's performance on a validation set or using cross-validation to check for overfitting or underfitting. Regularization techniques, such as early stopping or using smaller subsets of data for training, can also help in mitigating overfitting issues. We have done cross-validation to train the final model and get the avg score so that the model is not prone to overfitting.

References

[1] Silhouette Visualizer — Yellowbrick v1.5 documentation n.d., www.scikit-yb.org, viewed 22 July 2023, https://www.scikit-yb.org/en/latest/api/cluster/silhouette.html?highlight=silhouette

[2] scikit-learn 2019, sklearn.cluster.KMeans — scikit-learn 0.21.3 documentation, Scikit-learn.org

[3] www.datacamp.com. (n.d.). Python t-SNE with Matplotlib. [online] Available at: https://www.datacamp.com/tutorial/introduction-t-sne

[4] Team, G.L. (2020). What is Curse of Dimensionality in Machine Learning? [online] GreatLearning Blog: Free Resources what Matters to shape your Career! Available at: https://www.mygreatlearning.com/blog/understanding-curse-of-dimensionality/

[5] Karanam, S. (2021). Curse of Dimensionality — A 'Curse' to Machine Learning. [online] Medium. Available at: https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb

[6] SciKit-Learn (2009). 3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.21.3 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/cross_validation.html