Advanced Data Analysis
DATA 71200
Final Paper
Meriem Benarbia

Abstract :
Today the different Data, their size, comportment, and their resources are continually improving. Imagine handling Big Data from separate entries with a different format and having to normalize one by one their format knowing that at any time, you can face a new entry with a new type of form. You have to do a manual coding of this format and go through all the previous ones, which will incur a waste of time and cost without being sure that this work will be accurate. From that, the application of artificial intelligence (AI) Machine Learning (ML) came with different processes and models. ME in this project, I used an Insurance Dataset that I download from the Kaggle website. Dataset insurance has Ten features and 1400 raw. Some cleaning and preprocessing methodologies were used on this dataset, supervised learning, and unsupervised learning models under Python.
Keywords: AI, ML,features

Introduction:
Machine learning is an application of artificial intelligence (AI) that provides systems the ability to learn and improve from experience without being explicitly programmed automatically. Machine learning focuses on the development of computer programs that can access data and use it to learn from themselves. The process of learning begins with observations of data, such as examples, direct experience, or instruction, to look for patterns in data and make better decisions in the future based on the provided cases. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly. ML's advantages are the automation (no need to babysit the project every step of the way). Another two significant benefits are Handling multi-dimensional and multi-variety data besides Continuous Improvement since ML algorithms gain experience since it keeps improving in accuracy and efficiency.

Description of Dataset:
I understood from this course that a dataset is a structured collection of data generally associated with a unique body. This data are observations or measurements which are represented either as text, numbers, or multimedia.
As I worked before in banking firm, I wanted to choose a dataset which is close to electronic banking, I looked in all the website proposed on project 1, but I could not find one, so I did choose a dataset close to my previous field which is Insurance.
Insurance Dataset contains 1400 raw and 10 features which are:

- **Age:** age of policyholder.
- **Sex**: gender of policy holder (female=0, male=1)
- **BMI**: Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 25
- **Number of Steps**: average walking steps per day of policyholder
- **Number of children**: dependents of policyholder
- **Smoker** :smoking state of policyholder (non-smoke=0;smoker=1)
- **Region**: the residential area of policyholder in the US (northeast=0, northwest=1, southeast=2, southwest=3)
- **Insurance Claim** : yes=1, no=0
- **Date of claim.**
- **Charges**: individual medical costs billed by health insurance

In this Dataset, I tried to predict the Insurance *charges* by each user,

On python: in order to creation and analyze the dataset, two libraries are imported numpy as np and pandas as pd

```
data=pd.read_csv("insurance.csv")
#to print the full summary of the dataset, number of features and their DataType
data.info()
data.describe()
```

2) Detail what you did to clean your data and any changes in the data representation that you applied. Discuss any challenges that arose during this process.

Splitting the data
The first thing to do before processing the data cleaning is splitting the data between the training and testing set.The basic idea is to **divide** the dataset into two subsets – one subset is used for training while the other subset is left out for the test knowing that the performance of the model is evaluated on the test set.

In my case, I split my data set to an 80% training set and a 20% testing set (test_size=0.2).

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(data, test_size=0.2, random_state=42)
```

Data-cleaning:

An essential step in ML is Data Cleaning. There are several reasons why Data cleaning is necessary First of all: the data are picked up from different sources, so the data are not necessarily all accurate. Secondly, row data may have an incomplete record, noise values, outliers, and inconsistent data. Processing to analyze the data without cleaning leads inevitably to unreliable output and may negatively impact the predictive model.

In my project 1, a different techniques were used to clean my data. Below are those technique:

- **To a Numeric :**

When using data.info() at the beginning, which describes the data type of each feature, the feature charges were displayed as an object which was not accurate for a numerical feature. So, converting this feature to a numerical one was necessary: two steps were mandatory
1. Some number of charge feature have inside a comma ‘,’ so replacing this comma by space was mandatory

```
data['charges'] = data['charges'].str.replace(',', '')
```

2 .After replacing the comma, the conversion to numeric could be done

```
data['charges'] = pd.to_numeric(data['charges'],errors ='coerce')
```

- **Missing values:**

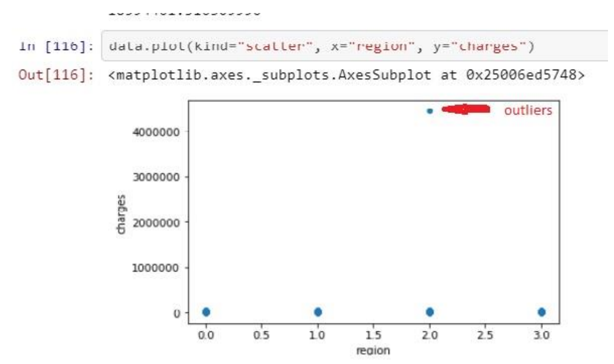
First, I had to check how many rows I am missing in each feature, so *I* used
`“df.isnull().sum()”`**: As a result**
BMI, Steps, Charges have few missing raw To Solve their problem; I replaced their missing values by their mean attribute, respectively. The syntax of the function used is

```
SimpleImputer(missing_values = np.nan, strategy="median")
```

The feature "**DateClaim**" has more than 70% missing raw, imputing its missing value can cause noise; therefore, dropping this feature seemed to be an excellent solution.

- **Noise value:**

The occurrences of noisy data in the dataset can significantly impact the prediction of any meaningful information. Noise data means that there is an error in data or outliers which deviate from normal. From the scatter plot of Charges and Regions feature, I observed one point that is distant from other observations



```
In [116]: data.plot(kind="scatter", x="region", y="charges")
Out[116]: <matplotlib.axes._subplots.AxesSubplot at 0x25006ed5748>
```

Looking at the left plot, we can observe that most of the data points are lying bottom side, but there is a point which is far from the charges usual point like the top right corner
since the outlier is the max value in this feature.Iuse
`“np.argmax(data['charges'])”`
and drop all the raw in which it belongs
`“data.drop([2], inplace=True)”.`

- **Data representation :**

Region feature has 4 categorical attributes: northeast, northwest, southeast, southwest, the four attributes are non-numerical and thus need to be converted to the computer can process them, many approaches exist to do this conversion, but from my side, I used label encoding approach to encode each region.
northeast=0, northwest=1, southeast=2, southwest=3

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data_cat = data["region"]
data_cat_encoded = le.fit_transform(data_cat)
```

- **Log transformation**

The transformation was done by the NumPy library, log function.

```
data_prepared["charges_log"] = np.log(data_prepared["charges"])
```

More details about this transformation are explained in the next topic "visualization understanding".

3) Discuss what you learned by visualizing your data

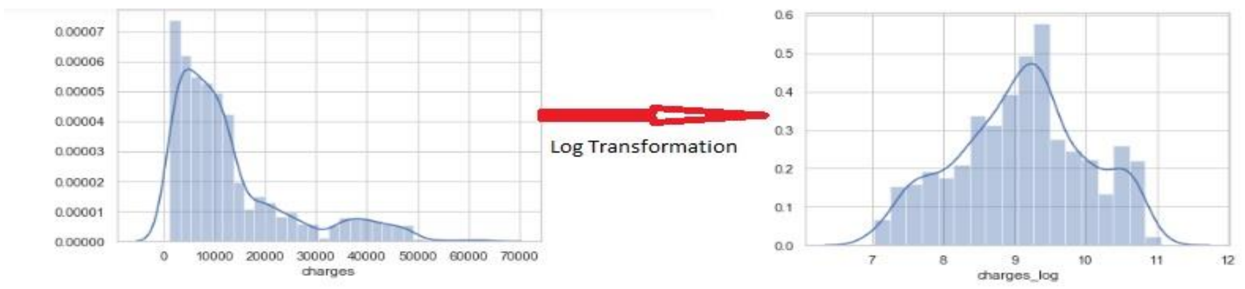
- **Histogram graph**

From the histogram graph, I could learn the class number of non-continuous data, and the distribution (shape) of continuous data.

```
data.hist(bins=50, figsize=(20,15))
plt.show()
```

The histogram of charges' feature and steps appear distorted or skewed to the right, which requires a transformation to get equal distribution.

The graph below illustrates the distribution before and after the log transformation

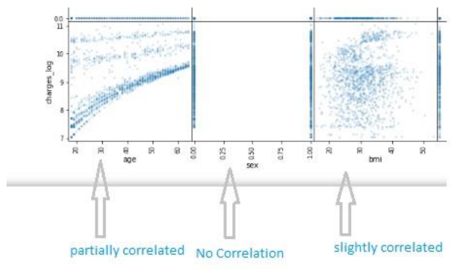


When running models without normal distribution, I noticed that accuracy was not that high. After that, therefore I conclude the normal distribution is westerly because it is easy for mathematical statistic and calculation to work with which will improve the accuracy of the models positively

- **Scatter plot matrix** :

```
From pandas.plotting and  import scatter_matrix
pd.plotting.scatter_matrix(data_prepared, alpha=0.2, figsize=(30, 30))
plt.show()
```

The scatter **plot matrix** allows us to visualize bivariate relationships between combinations of variables.
Each **scatters plot** in the **matrix** visualizes the relationship between a pair of variables, allowing many relationships to be explored in one chart.



As I wanted to predict the charge feature, it was important to visualize the relationship with this feature and the rest of the features.
From the graph at right, we can notice that there is no correlation between feature sex and charge which mean that sex feature does not contribute to predict the Y charges, therefore I dropped the sex feature

Supervised learning

A **Supervised learning** refers to a class of systems and algorithms that determine a predictive model using data points with known outcomes. The model is learned by training through an appropriate learning algorithm (such as linear regression, random forests, or neural networks) that typically works through some optimization routine to minimize a loss or error function.

4) Describe your experiments with the two supervised learning algorithms you chose. This should include a brief description, in your own words, of what the algorithms do and the parameters that you adjusted. You should also report the relative performance of the algorithms on predicting your target attribute, reflecting on the reasons for any differences in performance between models and parameter settings.

Model 1.KNN.K nearest neighbors :

```
from sklearn.model_selection import cross_val_score
reg2 = KNeighborsRegressor(n_neighbors=2)
```

1.Description: KNN is a simple algorithm that stores all available data point and classifies new cases based on the closest point, the number of the closest point that has to be checked is given by K preferably an impair (to avoid an equal classification).
Therefore the new case is classified by a majority vote of its neighbors
The K is parameters to adjust (the number of neighbors), to select the good K that's right for the data, I ran the KNN many times with different values of K and chose the k that decreases the number of errors while keeping the algorithm's ability to accurately make predictions when it's given data it hasn't seen before
in my case,I adjusted the k parameter K=default, K=2, k=9, also the parameter algorithm={'auto', 'ball_tree', 'kd_tree', 'brute'}
2. Performance of KNN on my dataset when predicting my target attribute "Charges"

Number of K	**Score Cross validation(1)**	**Sklearn.Metrics**	**Accuracy**
Default	0.35157847	MAE (2): 0.556747 RMSE (4): 0.79506 MSE (3): 0.632135 R-Squared (5): 0.30367	training set: 0.558 test set: 0.304
K=2	0.25223957	MAE: 0.573291 RMSE: 0.87547 MSE: 0.766460 R-Squared: 0.15571	training set: 0.765 test set: 0.156
K=9	0.36261472	MAE: 0.569569 RMSE: 0.799436 MSE: 0.639099 R-Squared: 0.29600	training set: 0.479 test set: 0.296

From the above table, we can notice that the KNN model for three number of k did not fit data, score (average of 5 cross-validations) express that fitting a new data with this model will inevitably fail. The metrics as MAE, RMSE, and MSE are high more than 50%, which explains that the difference between the predicted value and true value is high(high loss function).

(1)**Cross Validation** is **used to** assess the predictive performance of the models and to judge how they perform outside the sample to a new data set also known as test data. The motivation to **use cross validation** techniques is that when we fit a model, we are fitting it to a training dataset.
(2)MAE. Mean Absolute Error: It is the average of the absolute value of the difference between the measured "value "and "true" value (is used as the loss function)

(3) MSE. Mean Squared Error: is the average of the **squared error** between the measured value and "true" value that is used as the loss function
(4) RMSE. Root Mean Squared Error: is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values.
(2), (3) and **(4)** are the metrics for summarizing and assessing the quality of a machine learning, they are good measure of how accurately the model predicts the response. Lower values of (2), (3) and (4) indicate better fit, 0 means the model is perfect.
(5)R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination. In general, the higher the R-squared, the better the model fits the data.

Model 2 : Random Forest Regressor

```
from sklearn.model_selection import cross_val_score
Random = RandomForestRegressor(n_estimators=20, random_state=2, max_features=6)
```

Random forest is a collection of decision trees. As its name suggests, it is a Forest made of the root, trees, and leaves. In machine learning this algorithm operates by constructing a multitude of decision trees at training time to determine the final output. Higher the number of trees, better performance will be, but the execution will be slow.
In my project , I adjusted the number of trees which can be specified by n estimators parameter and I adjusted max_features parameter which is The number of features to consider when looking for the best split:

Adjusted features	Cross validation	sklearn.metrics	Accuracy
Default	0.77481053	**MAE**: 0.221954 **RMSE**: 0.39968 **MSE**:0.159747**R-Squared**: 0.7905	On training set: 0.959 On test set: 0.791

n_estimators=5
random_state=2
max_features=6
	0.77536579	**MAE**: 0.255278 **RMSE**: 0.43922
		MSE:0.19291**R-Squared**: 0.74712
			On training set: 0.954
			On test set: 0.747

n_estimators=20
random_state=2
max_features=6
	0.77714577	**MAE**: 0.232875 **RMSE**: 0.41422
		MSE:0.17159**R-Squared**0.77506
			On training set: 0.965
			On test set: 0.775

From the above table, we can observe that decision tree performs well on this dataset, either with the default parameter or with the adjusted one. The three parameter's value perform very nearly the same but n estimator =20 gave a better performance with only 1% compares to the others estimators. The three execution have at least %95 Accuracy on the training set, and a minimum of 74% of the test set and even the loss function is very low compares to KNN.

4. discussion about differences between the big difference in performance between KNN and Random forest:

The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set, but since as R-Squared was low that means the loss function" the difference between the actual value and the predicted value is high," therefore, KNN could not perform well, KNN could not found a big similarity between the testing and the training set, in contrast, decision tree does not save all the data point to compare it with the new cases, It breaks down a dataset into smaller and smaller subsets like a model, at the same time an associated decision tree with a series of yes/no questions helps to lead to predicting the new data point (like asking a sequence of queries about the available data we have until we arrive at a decision) that's why Random forest performed well than Knn in this dataset because of the big lack of similarities between the point.

5) Describe your experiments using PCA for feature selection, discussing whether it improved any of your results with your best performing supervised learning algorithm.

Principal component analysis (PCA) is a technique for reducing the dimensionality of datasets, increasing interpretability but at the same time minimizing information loss. scaling the data before using PCA can give a better results.

```
from sklearn.decomposition import PCA
pca2 = PCA(n_components=6)
pca2.fit(X_scaled)
X_pca2 = pca2.transform(X_scaled)
```

The total variance is the sum of variances of all individual principal components. Using `"y=pca2.explained_variance_ratio_."` helps to get the number of components needed to get a certain number of variance. In my dataset, it was interesting to observe that only six-component (the uncorrelated one) from 10 were needed to get 95% of the variance.

Discussing whether it improved any of my results with my best performing supervised learning algorithm.

Accuracy of Random Forest without PCA	Accuracy of Random forest with PCA
Accuracy on training set: 0.944 Accuracy on test set: 0.821	Accuracy on training set: 0.997 Accuracy on test set: 0.358

From the table above we can notice that Random forest with PCA fits the training set perfectly with almost 100% of accuracy, but unfortunately, it fails to fit the test set which expresses 35% of accuracy (overfitting); Therefore Random forest model with PCA did not improve the Random forest, but I observed that execution of Random forest with PCA was a little bit faster than the without PCA (this note should be considered when dealing with a very large dataset)

Unsupervised Learning

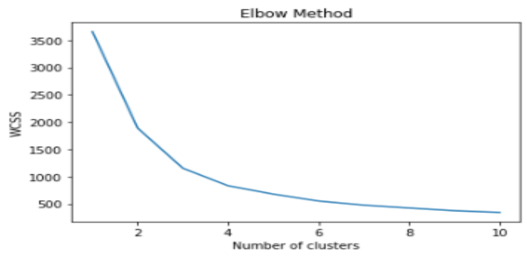
Unsupervised Learning is the area of Machine Learning that deals with unlabeled data. Broadly, it involves segmenting datasets based on some shared attributes and detecting anomalies in the dataset. It simplifies datasets by aggregating variables with similar attributes. The main goal is to study the underlying structure in the dataset. The two most common types of problems solved by unsupervised learning are clustering and dimensionality reduction.
Clustering As the name suggests, clustering involves dividing data points into multiple clusters of similar values. In other words, the objective of clustering is to segregate groups with similar traits and bundle them together into different clusters.
Silhouette Score: The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. Negative values indicate that those samples might have been assigned to the wrong cluster.

6) Discuss the results of using PCA as a pre-processing step for clustering. This should include a brief description, in your own words, of what the algorithms do.

- **Kmeans Algorithm :**

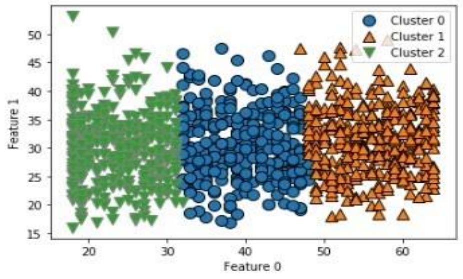
```
from sklearn.cluster import KMeans
kmeans_pca = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10,
random_state=0)
kmeans_pca.fit(X_pca2)
```

The K-means is a popular unsupervised machine learning algorithms, this algorithm identifies k number of centroids, which are used as the beginning points for every cluster, and then allocates every data point to the nearest cluster. The best number k can be identified by elbow method (As shown on the picture below)



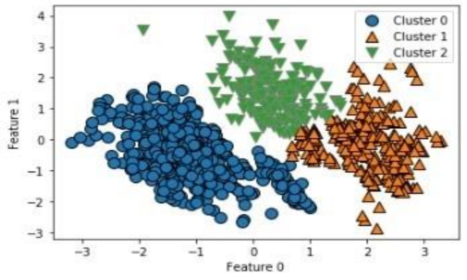
From the right graph ,K =3 is the best k for my dataset

- **Performance of Kmeans with and without PCA**



KMEAN WITHOUT PCA

silhouette_score=0.33



KMEAN WITH PCA

silhouette_score=0.48

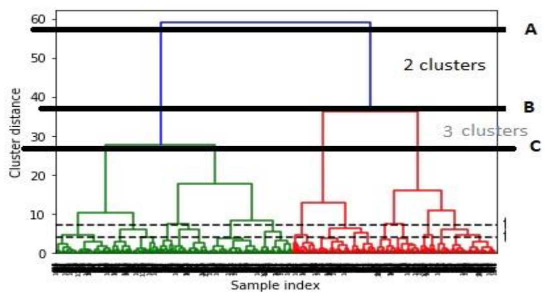
From the graph above we remark, that the KMEANS with PCA performs better than without PCA, noting improvement in silhouette score too, which means that more objects belongs to the right cluster, therefore in this dataset, dimensionality reduction improves the clustering task on the Kmeans algorithm.

- **Agglomerative Clustering Algorithm:**

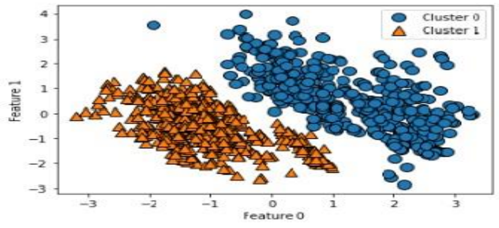
```
from sklearn.cluster import AgglomerativeClustering
agg = AgglomerativeClustering(n_clusters=3)
assignment1 = agg.fit_predict(X_train)
```

Agglomerative clustering is a “bottom up” type of hierarchical clustering. In agglomerative clustering, each data point is defined as a cluster. Pairs of clusters are merged based on their similarity as the algorithm moves up in the hierarchy. These clusters can be merged through a variety of methods. They all involve calculating dissimilarities between objects.

Dendogram express Height represents distance between objects/ clusters.we can use it too, in order to make a decision of the no. of clusters that can best depict different groups. The best choice of the no. of clusters is the no. of vertical lines in the dendrogram cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster.

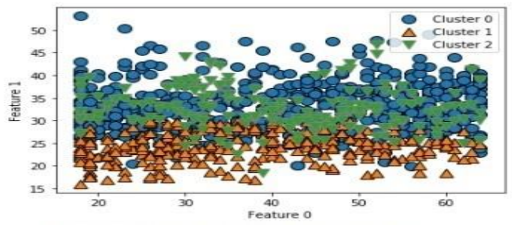
	In my dataset, the best choice of no. of clusters will be 2 as the black horizontal line in the dendrogram covers maximum vertical distance AB.

- **Performance of Agglomerative with and without PCA**



Agglomerative Clustering with PCA

silhouette_score= 0.46



Agglomerative Clustering without PCA

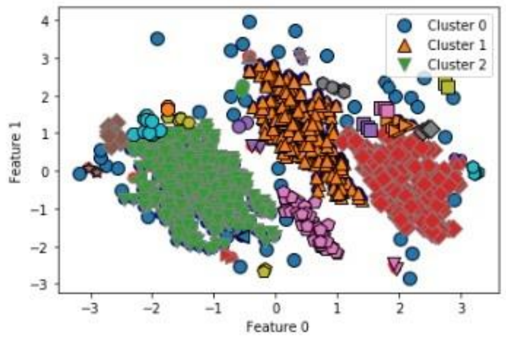
silhouette_score=0.37

From the above graph, we can notice that Agglomerative Clustering algorithm with PCA appears better than without PCA, the silhouette's measure with PCA express a higher than without PCA which explains that objects are more far away from the neighboring clusters than the Agglomerative Clustering algorithm without PCA

- **DBSCAN. density based clustering algorithm**

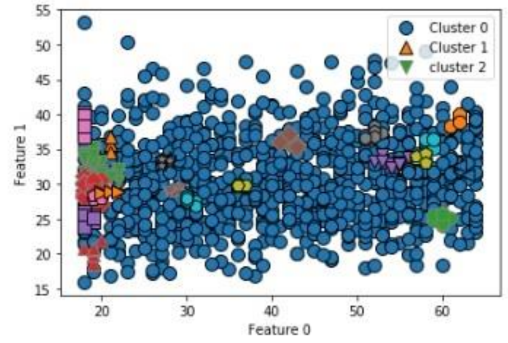
```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(min_samples=6,eps=0.5)
clusters = dbscan.fit_predict(X_pca2)
```

DBSCAN is a clustering method that is used in machine learning to separate clusters of high density from clusters of low density. DBSCAN algorithm requires two initial input parameters, namely Eps (the radius of the cluster) and MinPts (the minimum data objects required inside the cluster) which both have a significant influence on the clustering result



DBSCAN with PCA

silhouette_score=0.18



DBSCAN without PCA

silhouette_score=- 0.22

From the above graph , we notice that DBSCAN either with PCA or without PCA struggle to contort the data into a clear shapes in order to find similar clusters, I even tried to change EPS value many time but still the silhouette score did not get higher compares to the Kmean and Agglomerative clustering.

Conclusion:

7) Summarize what you learned across the three projects, including what you think worked and what you would do differently if you had to do it over.

Machine learning refers to a group of techniques that allow computers to learn from data; these techniques learn from this data to lead to a prediction. Therefore, this data needs a good cleaning to take off all the outliers and the noises to not lead a bad prediction. In supervised learning, it is important to find a model that can reduce the maximum possible the loss function to get very high accuracy, in my case, Random Forest worked very well with my data set. In unsupervised learning, clustering is a very sensitive task since we are working with unlabeled data, using PCA as I did. It can improve the performance of the model since it takes off the correlated features also reduce the time of execution. In my dataset Random Forest Regressor performed the best.

When going over all the projects at the same time, I wished having done more cleaning of data to better normalize the data. And If I have to do it again, I would surely choose a better dataset with more features to well observe each step of the ML and especially how PCA reduces dimensionality, and also to be able to observe good clusters results.

References:

[1] Hands-On Machine Learning with Scikit-Learn and TensorFlow_ Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media (2017)
[2] Introduction to Machine Learning with Python
[3] https://scikit-learn.org/stable/
[4] https://towardsdatascience.com/