**Project Report**

**Introduction:**

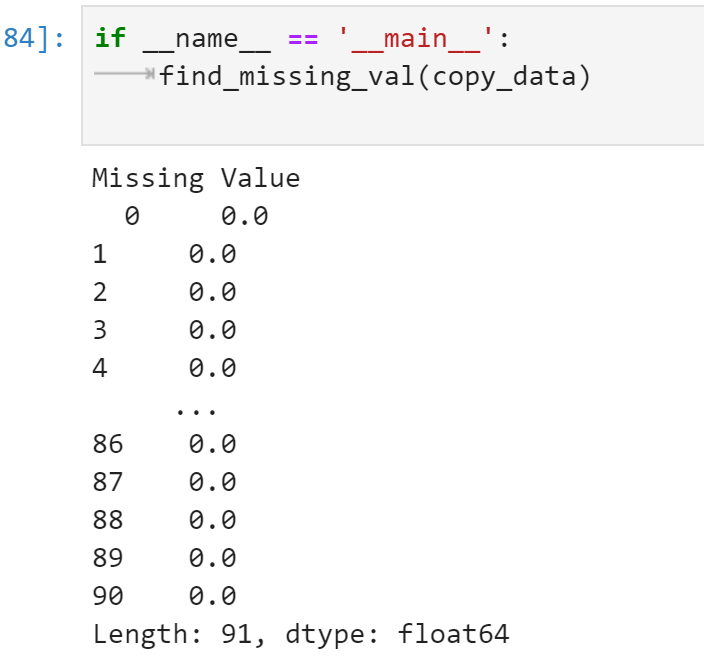
With the generation of an enormous amount of data generated every day, it becomes mandatory to find a way to store, manage, and process the data. This data can serve as a crucial building block to create a strong knowledge base and extract important information to use for further analysis and for various purposes. The data analysis finds its way in small scale applications like finding out user preferences based on past purchases to large scale applications in healthcare. Machine Learning models play an important role to bridge the gap of converting data to knowledge. Various regression and classification algorithms are used to achieve this.

One such dataset that can be extremely resourceful for various problem statements is the one million song dataset. It consists of attributes of a song like a timbre, tone, etc for a million songs. However, I am working on a subset of this dataset that is 515345 songs and 90 attributes. One column contains the release year of a particular song. The dataset is borrowed from UCI machine learning. Based on the attributes of a song, I aim to build a regression model that is able to predict the release year of the song. Every song has specific attributes that correspond to the type of music that is popular during that particular year or time period. Based on which it is possible to get the year for any other song given the features.

**Procedure:**

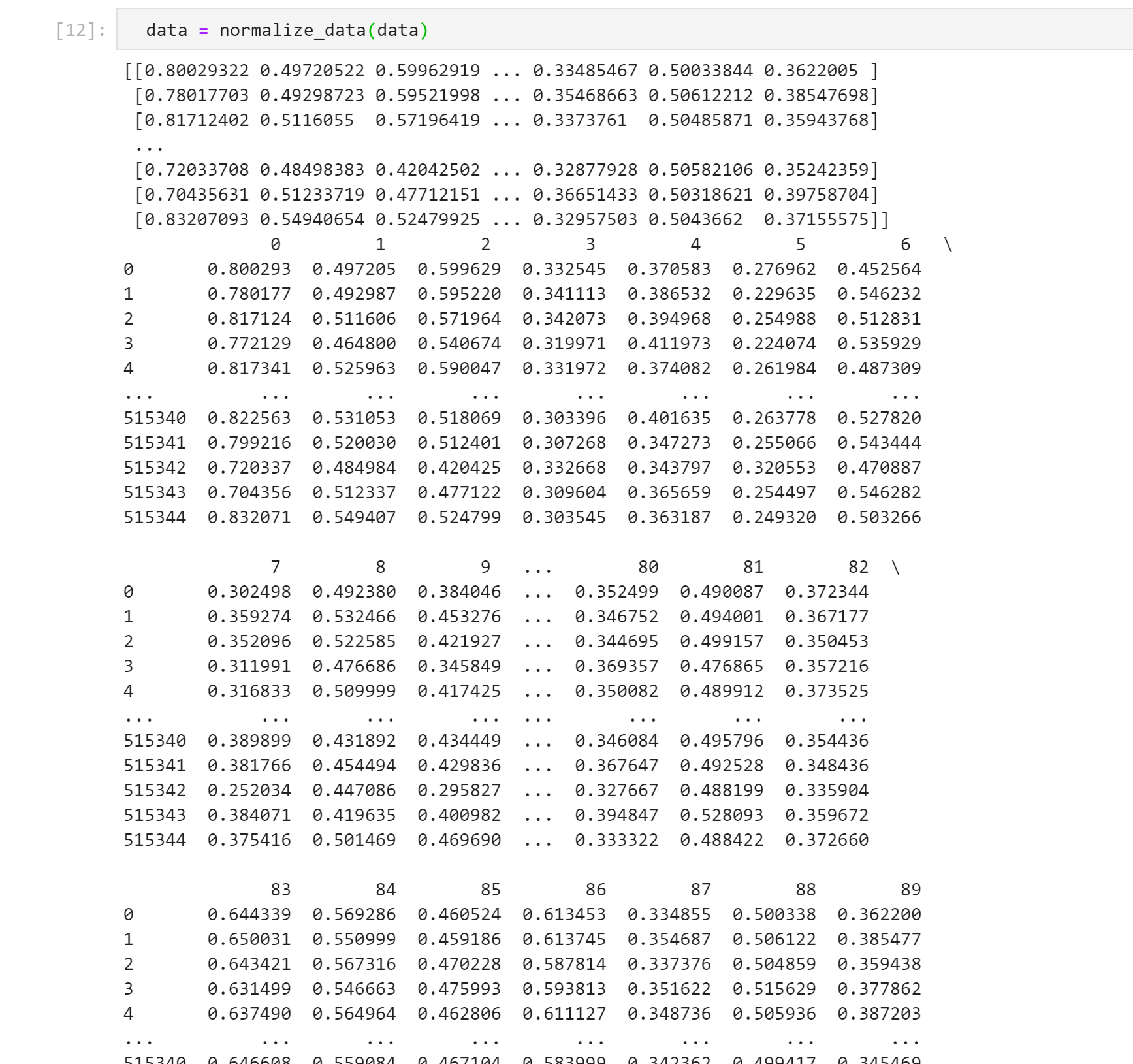
**Finding Missing Values:**

The dataset is contained in a .txt file (YearPredictionMSD.txt) and stored in pandas dataframe. As a preliminary approach to analysis, I decided to check if any of the attributes have missing values, NaN characters, or duplicates so that they can be handled by methods like forward-fill, backward-fill, or by averaging the previous and the next value for distributed data value. But the dataset is free from missing values. The function ‘find\_missing\_val’ takes the data frame as input and returns the number of missing values per column. As we can see that it is 0 for all the columns



**Data Normalization:**

The next step is to visualize the range of data for each attribute. For the dataset, the range was spread out between +60 to -40. This is a wide range and such a wide-range can make data processing and visualising difficult. So I have normalized the data into [0,1].



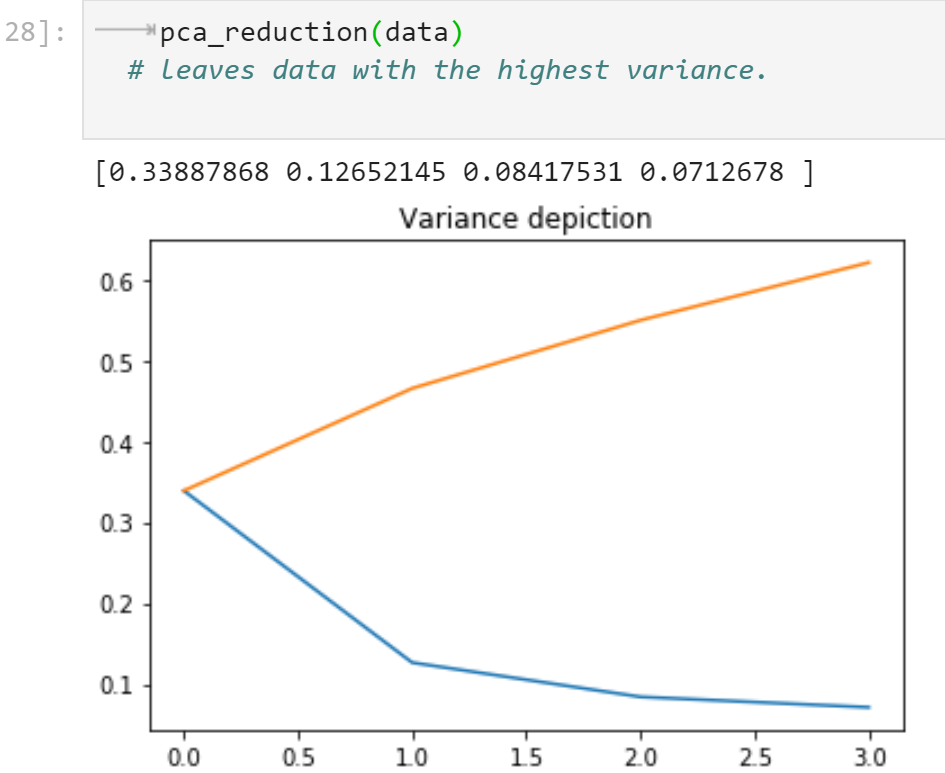
**Dimensionality Reduction (Obtaining the Variance-Ratio):**

Now that the data is normalized, I decided to analyze the important attributes from the dataset. The dataset has extremely large dimensions ([515345 x 90]). It is important to select the right attributes in order to build a more accurate model for prediction. We cannot reduce the number of songs since the more data available for training, the better are the results. Thus, I have tried to reduce the number of attributes so that I can remove redundant data, low variance data, or duplicate attributes.

The best way to find out how many attributes actually contribute to the variance is Principle Component Analysis. PCA is a dimensionality reduction technique that uses singular value decomposition to extract important features in the form of vectors. The first component has the highest variance-ratio and it decreases as we proceed to higher components. In order to get the best results and not leave out important features in the process, I have experimented with different values. The ‘pca\_reduction()’ function takes the data frame as input and returns the variance-ratio.

I observed that after a particular value of n\_components, the variance ratio stops increasing and becomes constant. I have used the PCA from sklearn and tuned the hyperparameter -- n\_components, for the best result. These are graphs of variance ratio per component (orange line) and cumulative variance ratio (blue line) for different numbers of components.

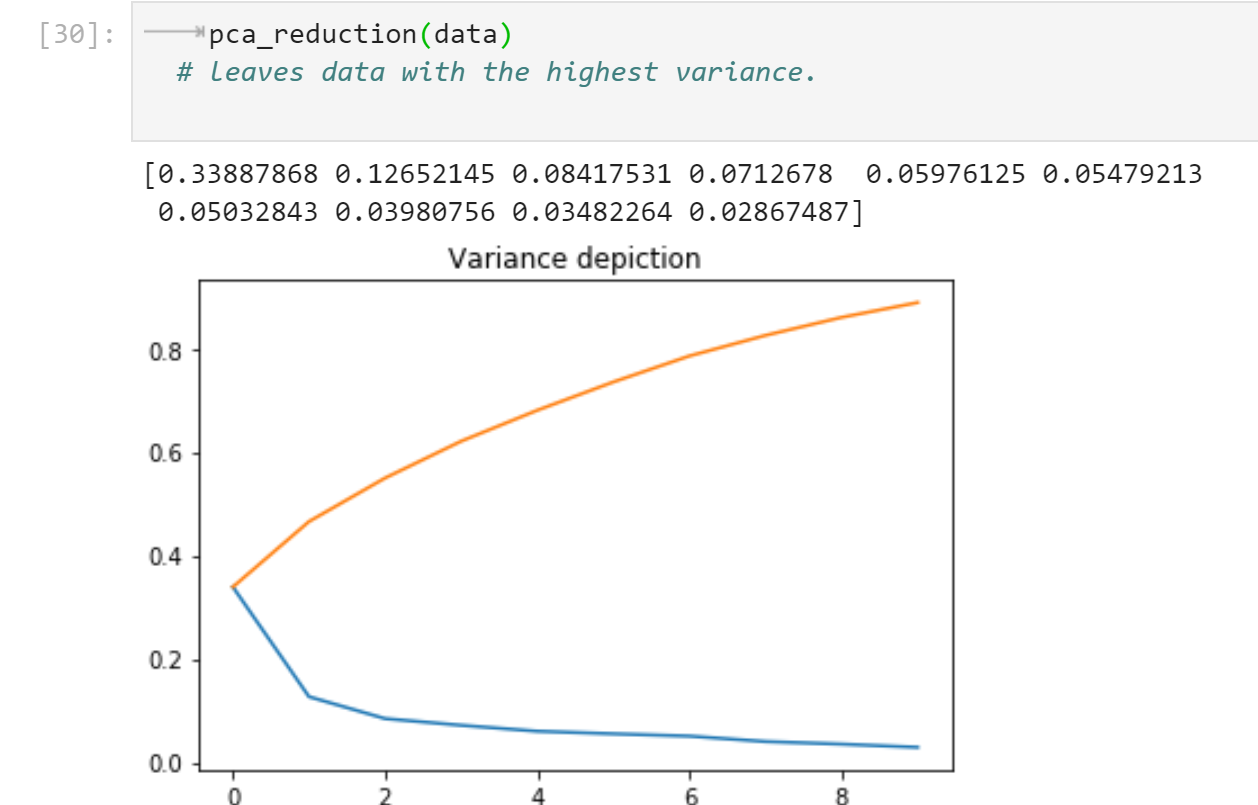
**When n\_components = 4**



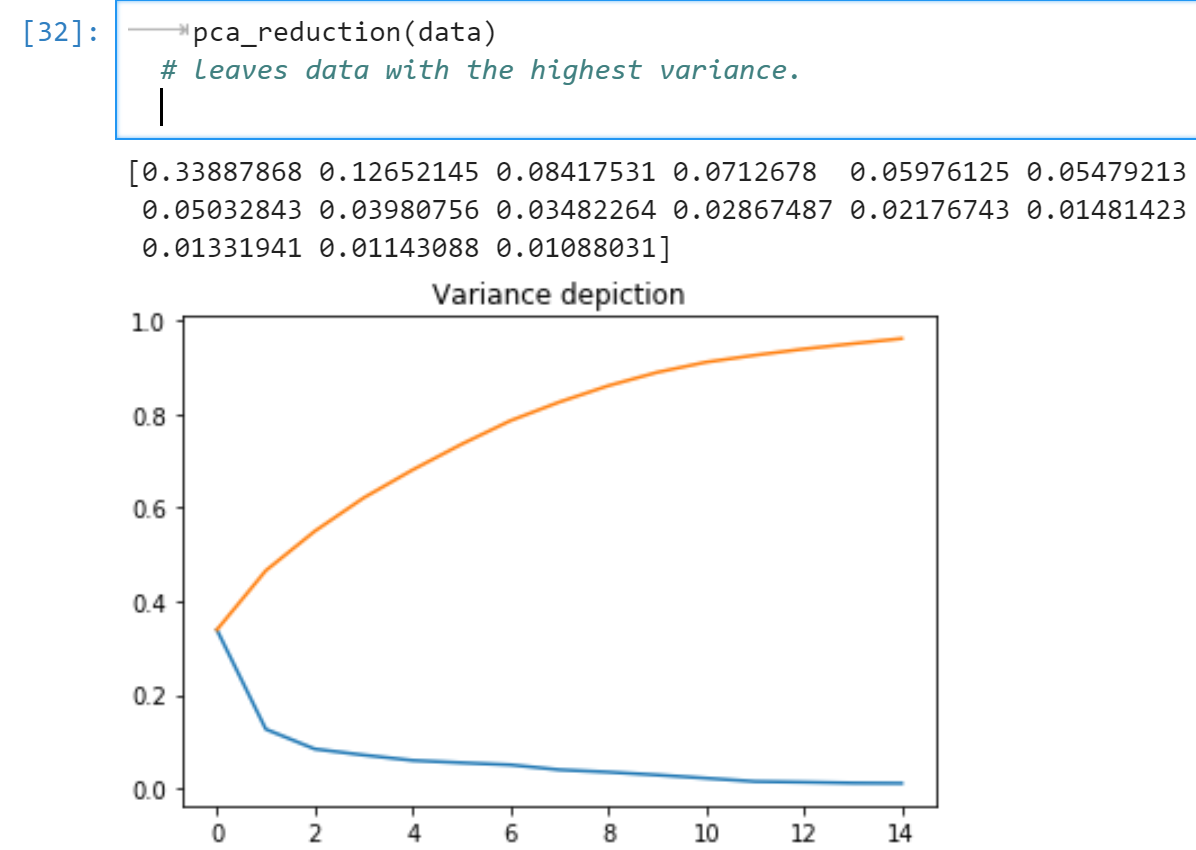
The values printed are the variance-ratios for each component. The first component has the highest value as expected and it decreases further. The maximum variance obtained is about 60%.

**When n\_components = 10**

The maximum variance obtained is more than 80%. Since the number of components is more, we are able to observe more variance, and hence, there is such a big gap between the above result and this one.

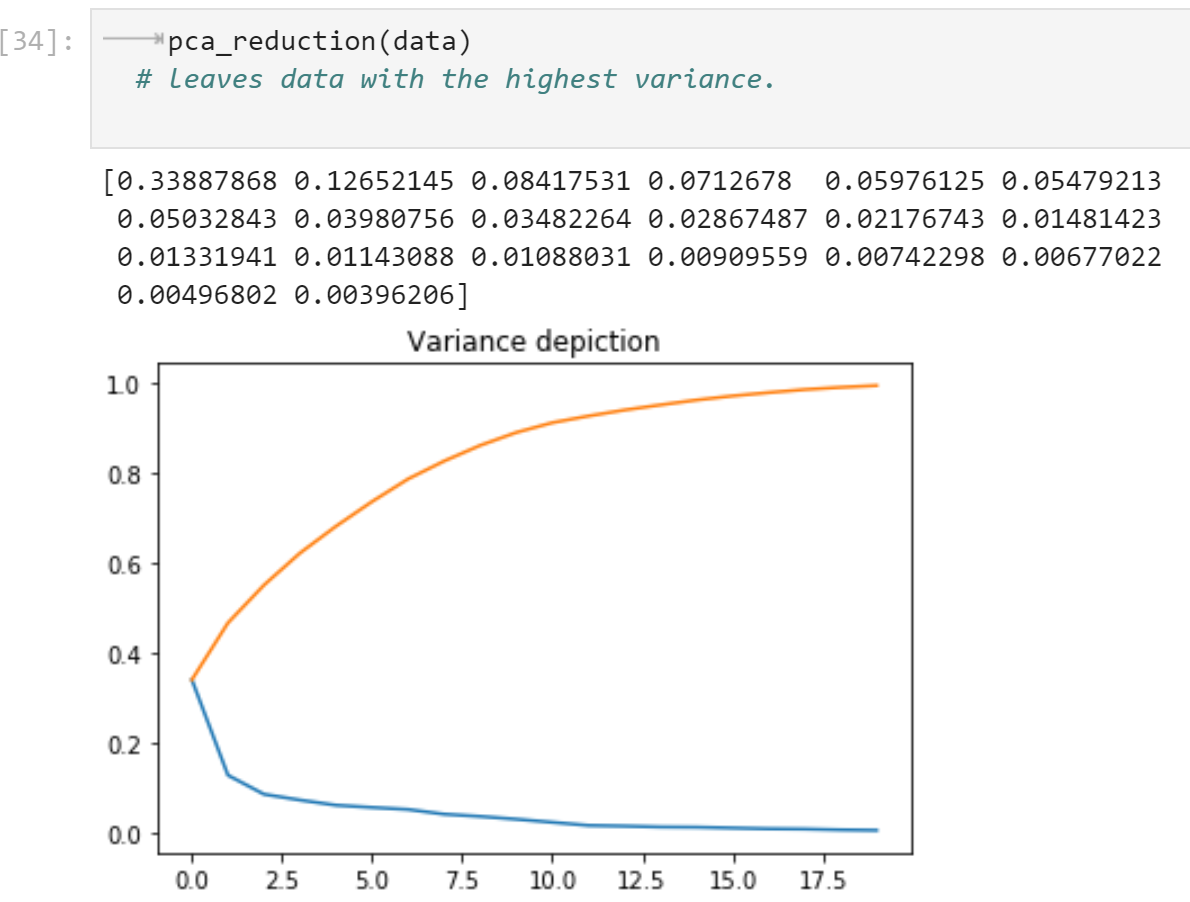


**When n\_components = 15**



Here too we can say the considerable difference in the variance-ratio. It is almost close to 1.

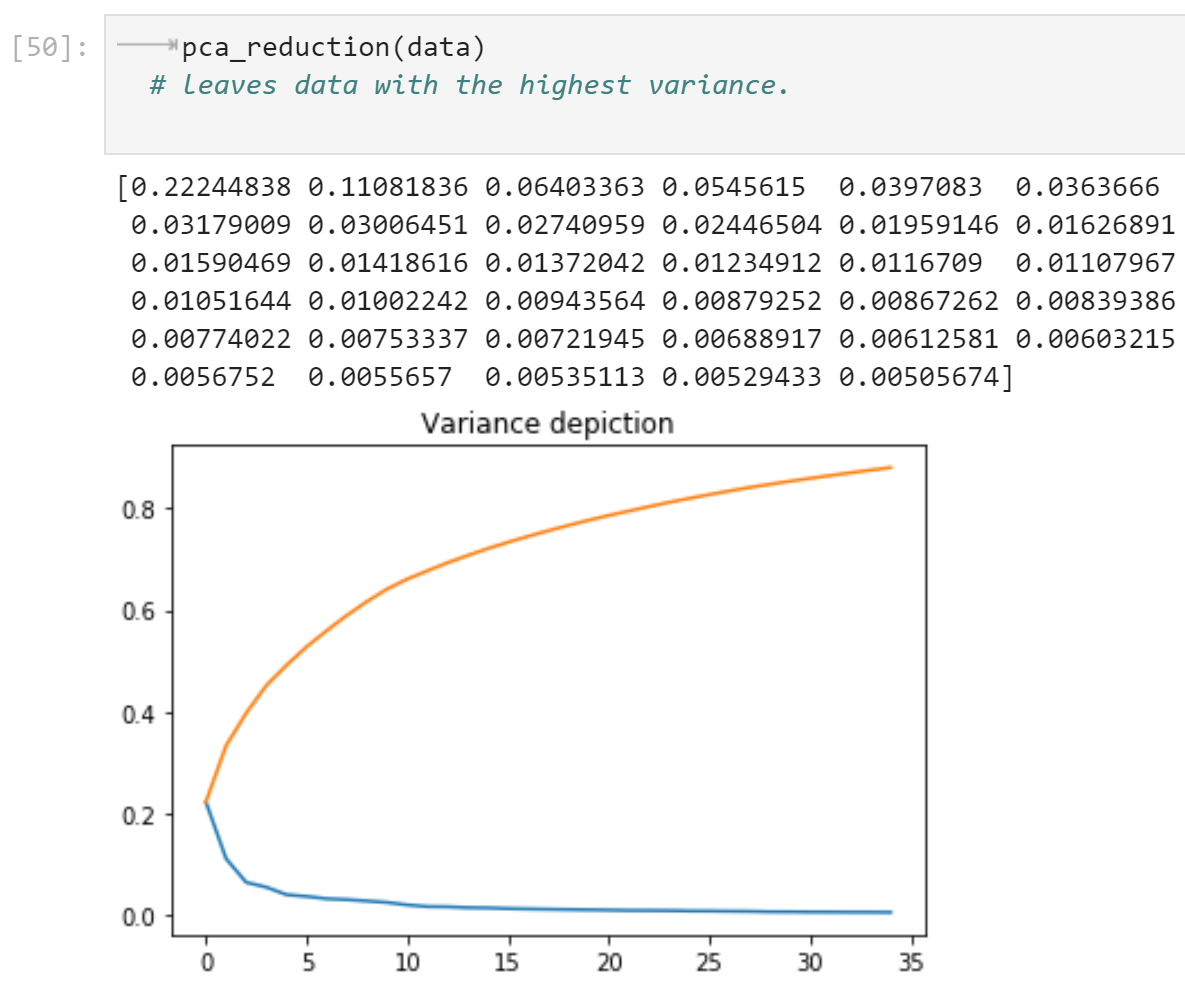
**When n\_components = 20**



As opposed to the other graphs, this one does not show much difference. So we can now conclude that the variance-ratio will not show significant change. I experimented with values of n\_components = 25, n\_components = 30, and n\_components = 35

However, after 20 it is more or less the same. But it is important to not leave out any important features just to eliminate the dimensions. So I decided to go with the value of **n\_components = 35**. Even if there are insignificant features, they will be reduced further in the process.

This is the visualization of the final value of n\_components.



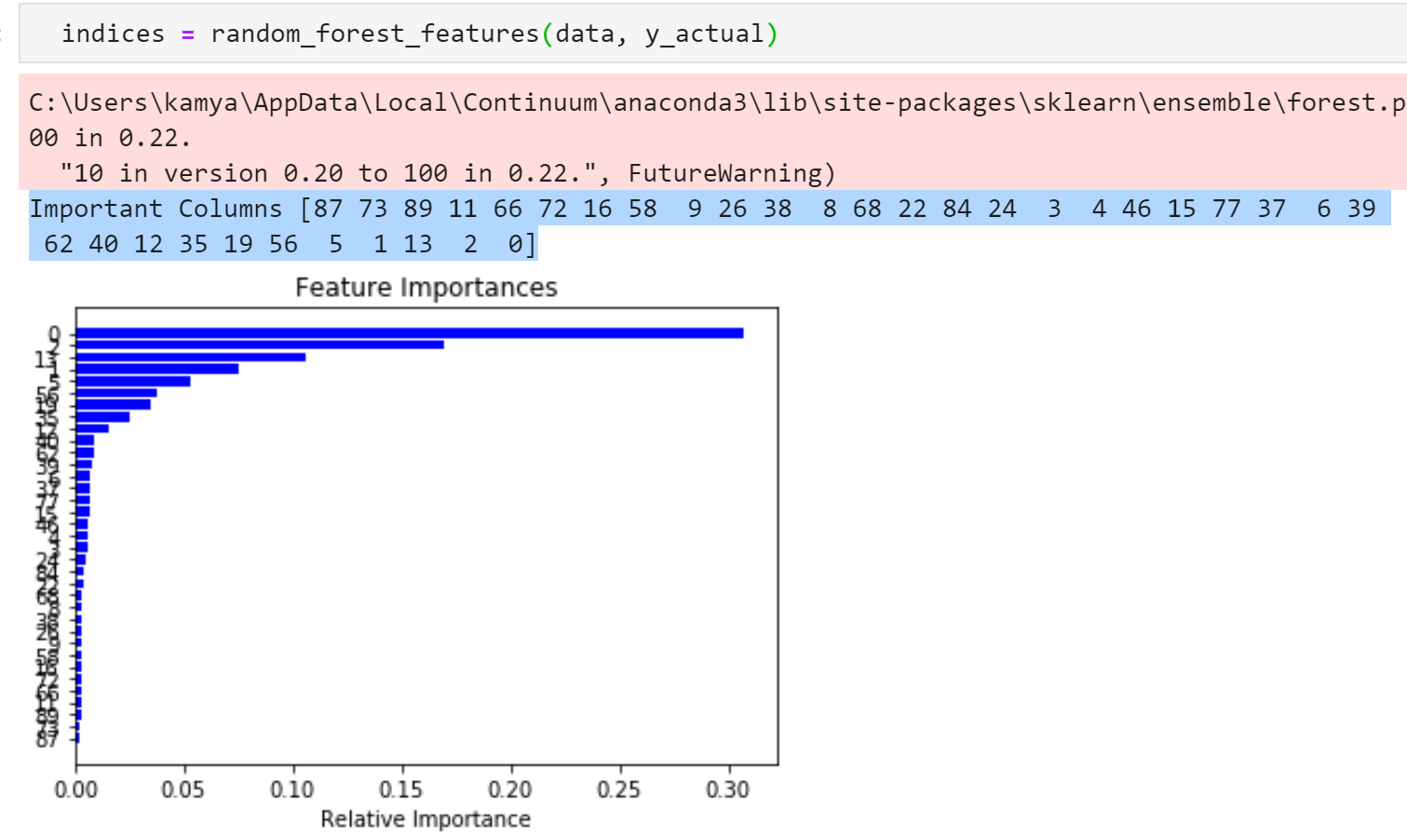
**Feature Pruning/Extraction:**

So I have 35 attributes that have high variance and can be used for prediction using a regression model. However, PCA only tells us maximum variance-ratio distributed over the attributes. To find out which attributes/column numbers that actually carry this variance, I have used random forest.

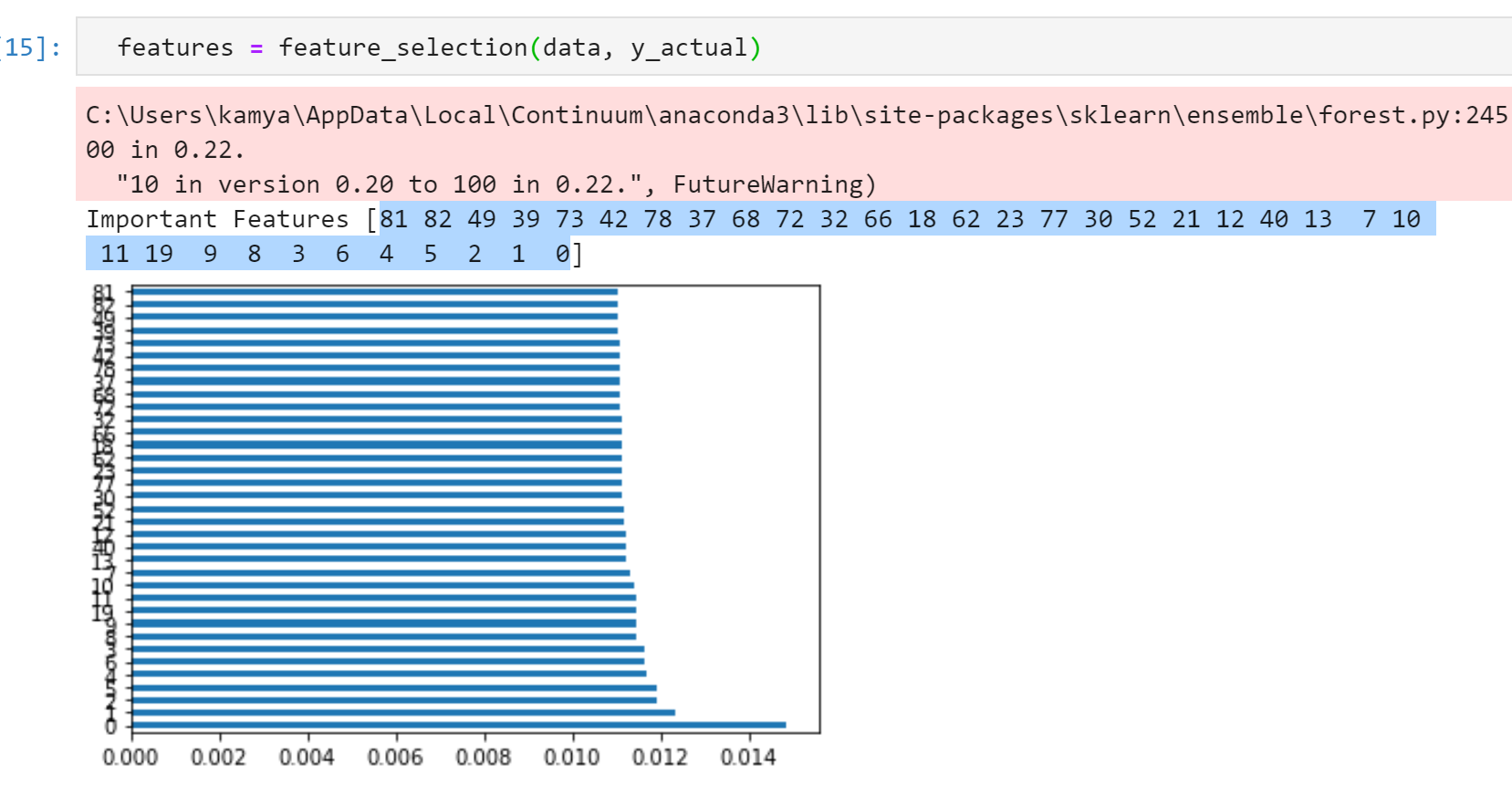
The hyperparameters I have used are:

1. Number of trees = 10
2. Number of features = 35 (As obtained from PCA)

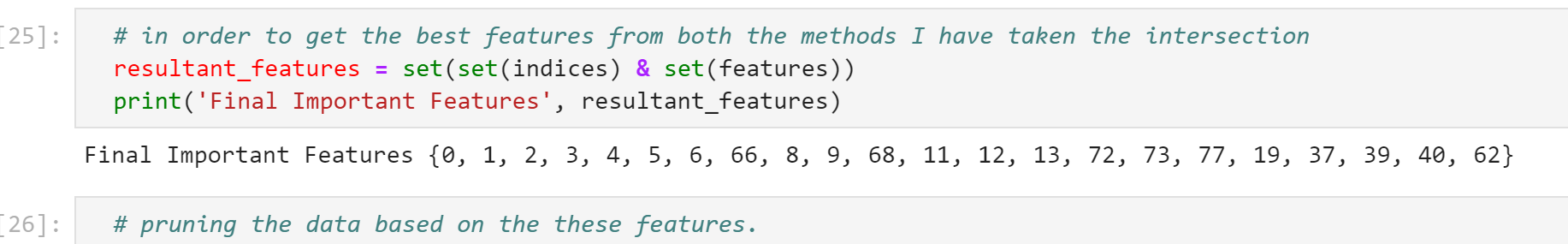
The ‘random\_forest\_features()’ function takes the data frame as an input and returns an array of column numbers/features that are most relevant. In this image, the array is in increasing order of importance (0 is the most relevant feature and 87 is the least important).



I have additionally used another feature selection tool, ExtraTreesClassifier() from sklearn.ensemble. This is to improve the probability that the selected features are most relevant and carry the most variance. The ‘feature\_selection()’ function does exactly that.



As we can see that the features obtained from Random Forest and the Extra Trees are not exactly the same. I have taken the intersection of these classifiers and pruned the features accordingly. These are the final features.

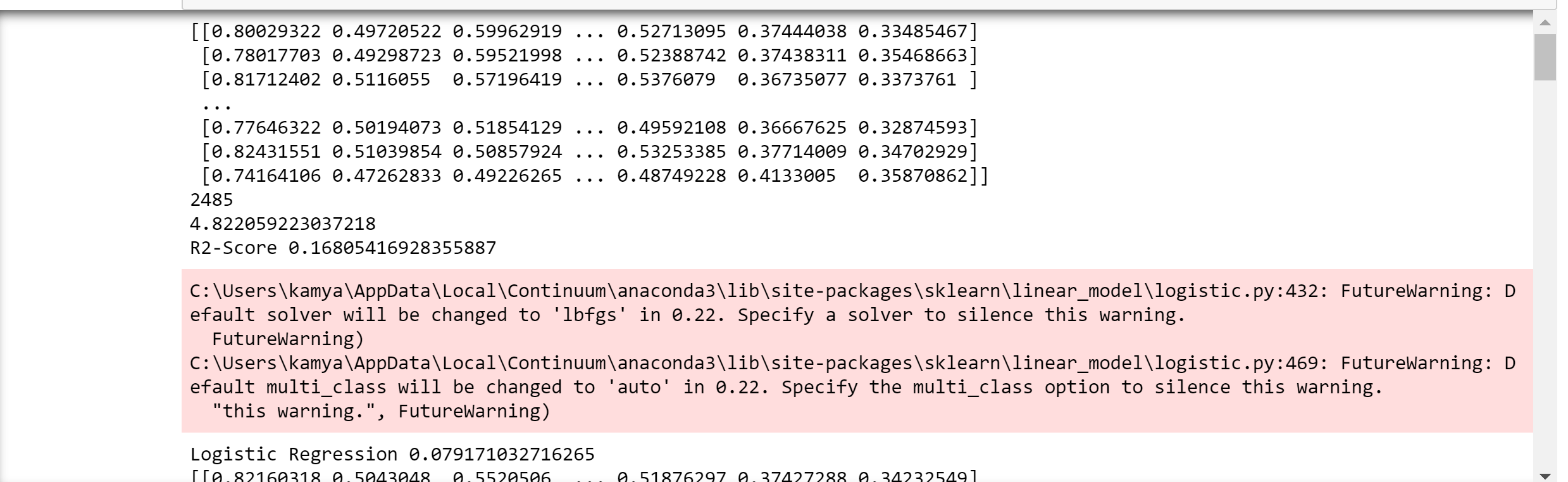


**Building the model:**

To get the best results I have decided to experiment with these regression models:

1. Linear Regression
2. Logistic Regression

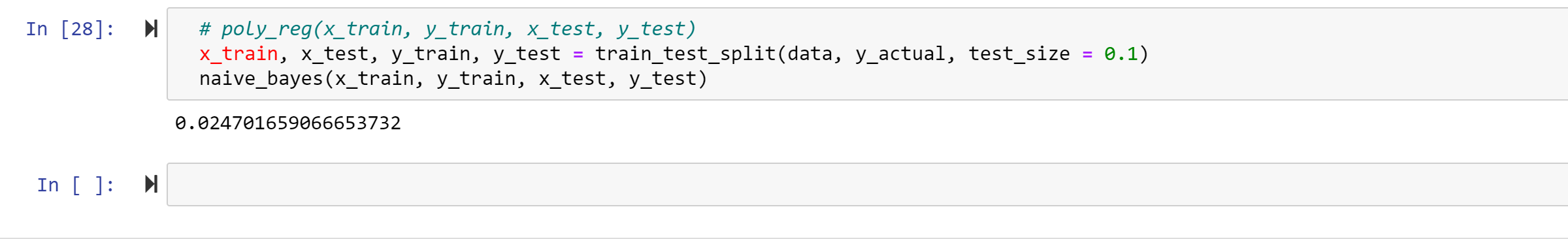
However, I did not get good results from both the models. I experimented with a train\_test\_split of 90 and 10. The main reason is that the predictions that I receive are in the form of float. For example, y\_predict contains values like 1997.456, 2001.99, and so on. However, the ground truth labels consist of a particular year. Even if I try to take the integer value for the same, it will not be accurate at all since we don’t know whether to take the floor or the ceil of the float.



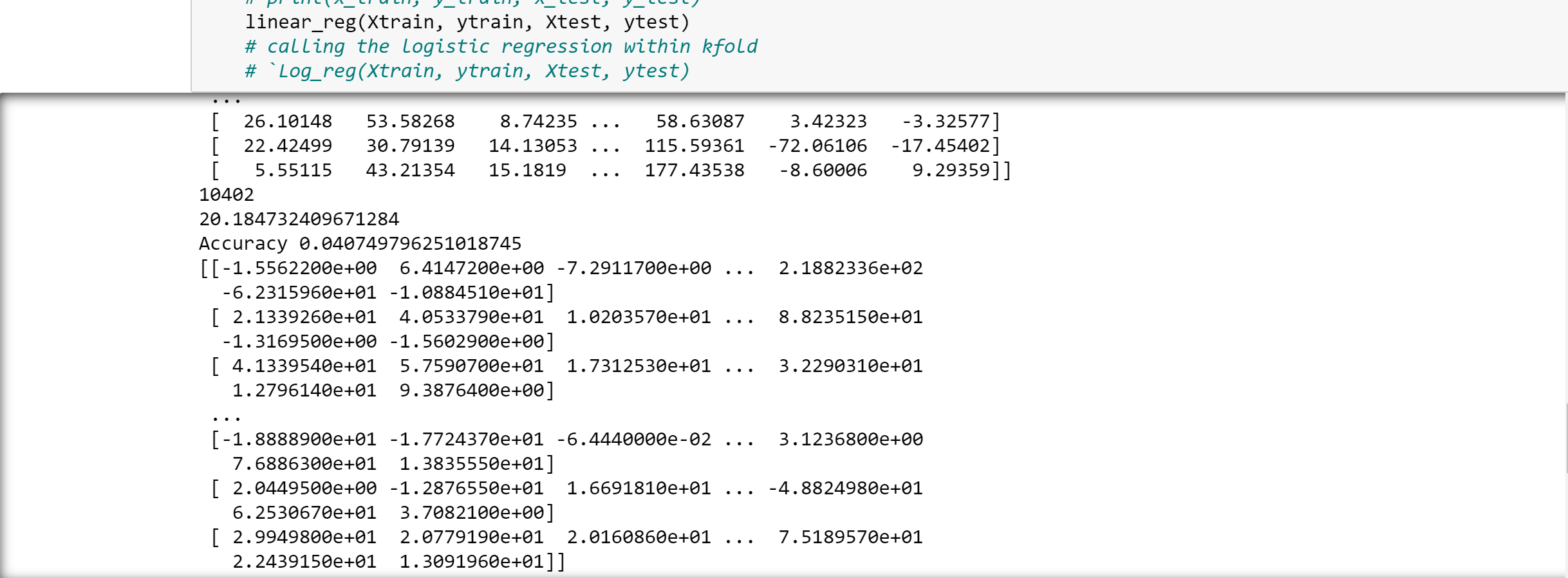
To improve the accuracy I have done k-fold cross-validation with the value of k = 10. But it could barely improve the numbers.

So I decided to move on to the classification algorithms. I decided to treat every year as a separate class. But it did not work well with the classification algorithm as well. For experimentation, I have used

1. Naive Bayes
2. KNN



Experimented Linear Regression while allowing a prediction gap of +-2 years. The maximum accuracy obtained was 20% (much better than before)



**Important Inferences:**

From the above experiments, I have made the following conclusions about the dataset and models.

1. The dataset consists of ground truth being an individual year (1997, 1996,..). When the regression model is applied, the predicted values for the years are in the form of 1996.3456, 1197.89777,... which means that they do not directly match the year(ground truth label). If I consider the int of these values, they randomly round up to the closest integer. However, in order to match to the ground-truth label, we do not know whether to consider the floor or the ceil.
2. Also, on viewing the values, I observed a considerable difference, for example, for a true label of 1996, the value predicted was 1994.3456. In cases like these (majority ones), rounding off wouldn’t work. But for a range of so many years, almost 90 years **(1922 to 2011**), a difference of +-2 years using a simple linear regression model is considerable.
3. It is possible to get a better accuracy as and when we increase the year interval, that is if we allow a deviation of 2 years or more. The more the gap is, the better is the accuracy as shown in the very last image.
4. Similarly with the classification algorithms, if we consider every year as a separate class, the accuracy would be less as compared to a group of years.
5. However, there might be more to this dataset than just type of model. According to me, the attributes consist of values pertaining to timbre, (12 attributes containing timbre average and 78 attributes containing timbre covariance) that must have a lot of correlation and affect each other while determining the year. Perhaps a more complex model is needed to extract this correlation, calculate the covariance, and extract the relevant features.