RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

PROJECT REPORT

Analysis of PCA Techniques for Brain Tumor Classification using Machine Learning Models

Prepared by

Vishnuram Jatin Bangaru (vb440)

Ankeeta Priyam (ap2213)

Smrithi Shankar (ss4350)

Varun Gandikota (vg411)

Table of Contents

ABSTRACT

Dimensionality reduction is a critical aspect in improving the accuracy of machine learning models and avoiding overfitting. In recent years, various techniques have been developed to address this issue, and we aim to explore these techniques with respect to principal component analysis (PCA) and its different variations. We will use a brain tumour dataset for our analysis and compare the performance of various PCA methods for dimensionality reduction. Finally we will employ various machine learning models to classify the tumors and conduct a comprehensive evaluation of each approach's efficacy.

The objective of this study is to provide insights into the effectiveness of different dimensionality reduction methods for enhancing the accuracy of machine learning models, particularly in the context of medical data analysis.

INTRODUCTION

We obtained the Brain Tumour Dataset from Kaggle, which contains images classified into three types of brain tumours: Glioma Tumour, Meningioma Tumour, and Pituitary Tumour. Glioma Tumour originates from the glial cells that support and nourish neurons in the brain. Meningiomas develop from the meninges, the protective membranes surrounding the brain and spinal cord. Pituitary Tumours are located in the pituitary gland, which produces hormones that regulate bodily functions.

Brain tumour classification is the process of categorizing brain tumours into different types based on their characteristics, such as tumour size, location, and cell type. The symptoms of brain tumours depend on their location, size, and type.

As part of this project, we will be extracting features out of the Images using a technique called Graycomatrix which calculates the statistical relationship between the intensity values of adjacent pixels in an image, which can be used to describe the texture of the image.

Upon feature extraction, we will use dimensionality reduction techniques of PCA, Sparse PCA, Kernel PCA on the features and later classify using ML models.

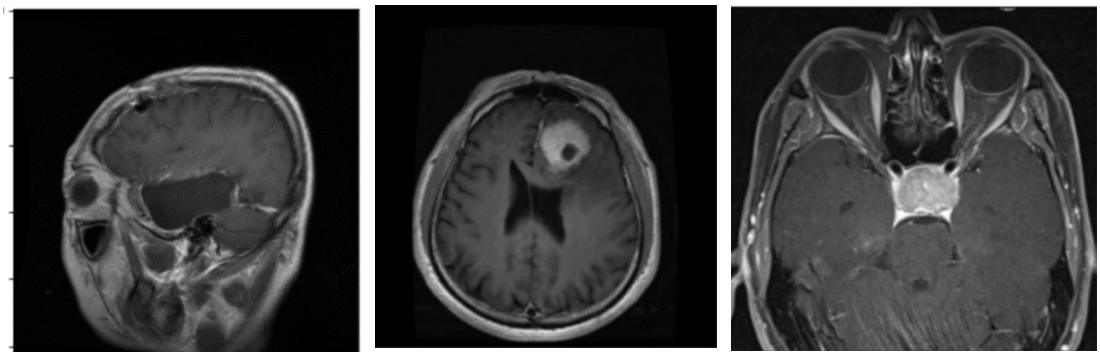
We will compare the different types of PCA using plots and how the proportion of variance is calculated and the number of components needed. Each of the PCA are unique in their style which are derived from ML concepts like Regression, SVM.

For this project, we will be using a set of ML algorithms like Linear Models, Ensemble models, boosting models etc.

MATERIAL AND METHODS

❖ Dataset

We have taken the Brain Tumour Classification dataset from Kaggle which consists of 3265 images consisting of Glioma, Meningioma and Pituitary Tumour along with images having no tumor. The categories are equally balanced having around 900 images. For the scope of the project, we have removed classifying No Tumour images and will concentrate on just identifying the tumour images. We have encoded the Tumor or the output as 0(Glioma), 1(Meningioma) and 2(Pituitary) in the order as seen below



❖ Feature Extraction

We have come across a unique implementation of classifying images rather than using Deep Learning CNNs which is extracting features from a method called Gray Level Co-occurrence Matrix. It calculates the statistical relationship between the intensity values of adjacent pixels in an image, which can be used to describe the texture of the image. The GLCM is a matrix that represents the frequency of occurrence of pairs of pixels with specific values and spatial relationships. The matrix is created by analysing the co-occurrence of gray levels at various spatial distances and orientations within an image. For our set of images, we have extracted the following features:

- **Contrast:** Measures the local variations in the gray-level co-occurrence matrix. Higher contrast values indicate a greater difference between the co-occurring gray-level values and therefore greater textural heterogeneity.

- **Dissimilarity:** Measures the average difference in the gray-level values between adjacent pixels. Higher dissimilarity values indicate greater textural heterogeneity.

- **Homogeneity:** Measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. Higher homogeneity values indicate a more homogeneous texture.

- **Energy:** Measures the sum of squared elements in the GLCM. Higher energy values indicate a more homogeneous texture.

- **Correlation:** Measures the degree of linear dependency between the gray-level values in the image. Higher correlation values indicate a more linear relationship between the gray-level values.

Few Other Properties:

- **Distances [1, 3, 5]:** This parameter specifies the distances between adjacent pixels to be considered in the GLCM calculation. The code calculates the GLCM for distances of 1, 3, and 5 pixels.

- **Angles [0, np.pi/4, np.pi/2, 3*np.pi/4]:** This parameter specifies the spatial orientations between adjacent pixels to be considered in the GLCM calculation. The code calculates the GLCM for four different angles: 0 degrees, 45 degrees, 90 degrees, and 135 degrees.

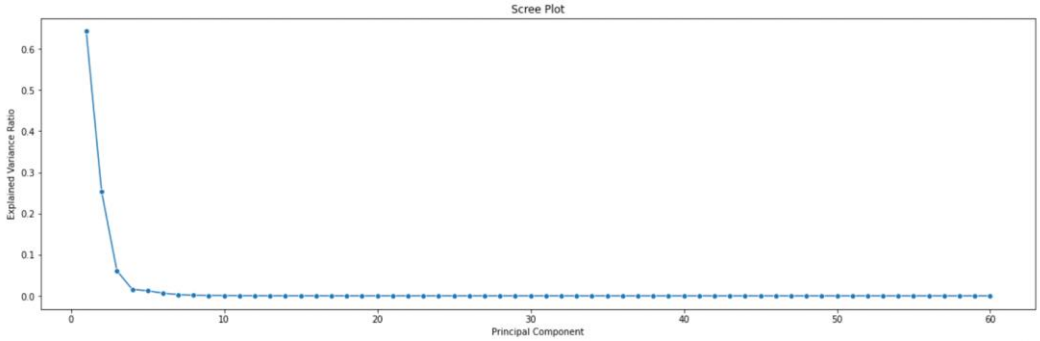
For each image, the GCLM matrix is calculated using the properties we defined and then extract the statistical measures from the Matrix.
The code extracts 5 different statistical measures (properties) from the Gray Level Co-occurrence Matrix (GLCM) for 3 different pixel distances and 4 different angles, resulting in a total of 5 x 3 x 4 = 60 features per image below

contrast_d1_a0	dissimilarity_d1_a0	homogeneity_d1_a0	energy_d1_a0	correlation_d1_a0	c
31.557313	65.589738	37.748452	66.758108	222.662708	
32.984329	69.372919	37.095592	65.085413	229.143871	
29.623800	49.832461	25.459477	55.787332	207.524688	
34.920377	74.073747	39.759720	69.379563	248.795121	
59.328737	101.134217	44.569311	97.633273	386.750015	
...	...	...	...	...	
40.142452	75.919486	34.718062	67.472398	258.259700	
60.759028	89.116134	33.292663	90.551874	372.954364	
41.557340	86.068547	40.331041	72.888795	300.340638	
42.480423	89.045714	50.834049	89.084869	287.383119	
65.247405	107.119672	52.032607	115.834173	404.450470	

❖ PCA

We now begin to reduce the dimensionality of the 60 features in our dataset. The first technique is the common PCA. Principal Component Analysis is an unsupervised technique which consists of finding components which preserves the variance of the original features. In PCA, we calculate the scores which is a linear combination of all the features. Generally, the first PCA is in the direction to the feature vector space while the second component is orthogonal to the feature space.

1. First the features are scaled to have 0 mean and 1 standard variation.
2. Calculate the eigenvectors and eigenvalues of the covariance matrix: The eigenvectors are the directions in the new coordinate system, while the eigenvalues represent the amount of variance explained by each eigenvector.
3. We then choose the number of principal components such that the variance explained is equal to the variance explained of all the features together. We generally end up reducing features to 2-5 components. As part of our analysis, we have performed PCA on the 60 features and found that 5 components explain 97% of the variance. The variance and proportional variance explained can be found by plotting scree plot:



❖ Sparse PCA

Sparse PCA is a variation of PCA which originates from the Lasso L1 Norm used in Regression to reduce the variance. It aims to find a small number of principal components that are not only able to capture the most significant variations in the data but also have sparse loadings, meaning that they only depend on a subset of the original variables.

In standard PCA, each principal component is a linear combination of all the original variables, making it difficult to interpret the relationship between the components and the original variables.

Sparse PCA addresses this issue by constraining the principal components to have sparse loadings, which leads to a more interpretable representation of the data. Sparse PCA finds the sparsity by adding a penalty parameter to the objective function.

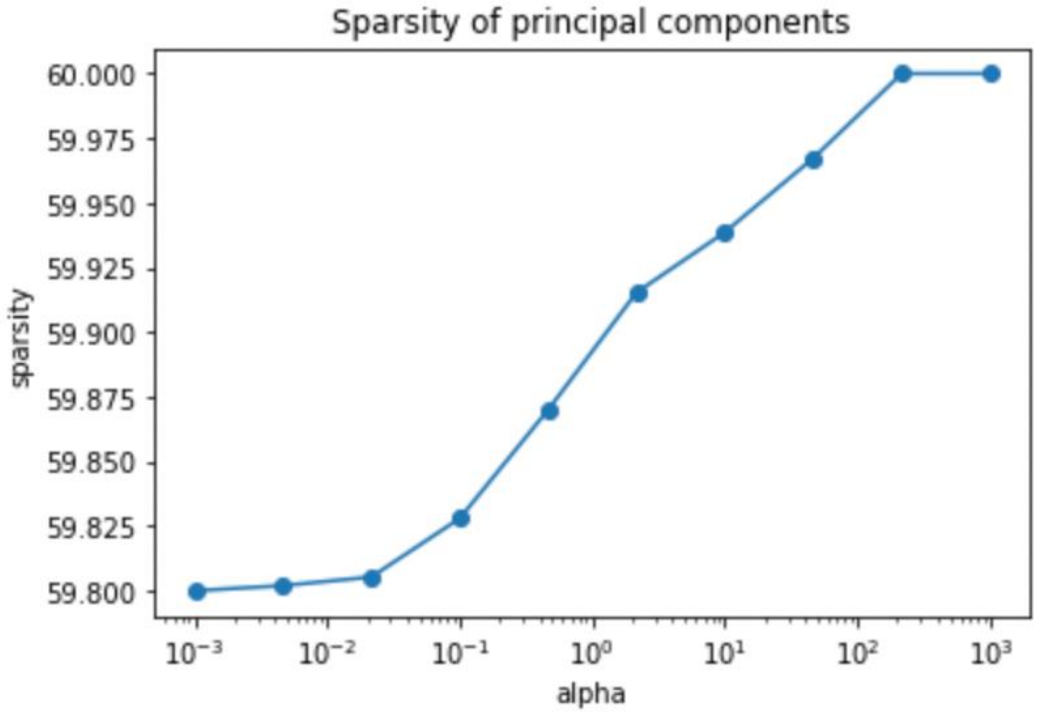
The Sparse PCA sklearn function has a parameter to control the Ridge Alpha and the Lasso Alpha like the elastic net. The results are components which uses a subset of features which are important and makes few scores as zeros similar to Lasso and Ridge Regression:

$$\hat{\beta} = \arg\min_{\beta} \|Z_i - \mathbf{X}\beta\|^2 + \lambda\|\beta\|^2 + \lambda_1\|\beta\|_1,$$

It was quite interesting to find the proportion of variance explained in Sparse PCA as the components unlike standard PCA are not orthogonal. However, after quite a research, we came across a formula to calculate the variance explained from a paper:

In general, the captured/explained variance in a component model with one sparse mode should be measured as $\text{trace}(PT^TTP^T)$.

T are the scores after fitting the Sparse PCA and P is the loadings. We have also observed that as we increase the alpha (regularization parameter), the sparsity d(sparse loadings). Alpha is used to control the strength of the penalty term introduced on the objective function. Increasing alpha too much can lead to over sparsification, where the principal components become too sparse and may lose important information.



❖ Kernel PCA

While PCA and Sparse PCA are good for constructing components when the data is linearly separable, Kernel PCA is good when the data is non-Linear and leads to constructing non-linearly separable components.

Kernel PCA originates from the idea of SVM to use kernels. We map the data to a higher dimensional space and find components in the new data space. Basically, in the higher dimensional space, the data becomes linearly separable where we can then do

a linear combination of data. Like the SVM, we have different kernels which transform the non-linear data into higher dimensions such that they become linearly separable.

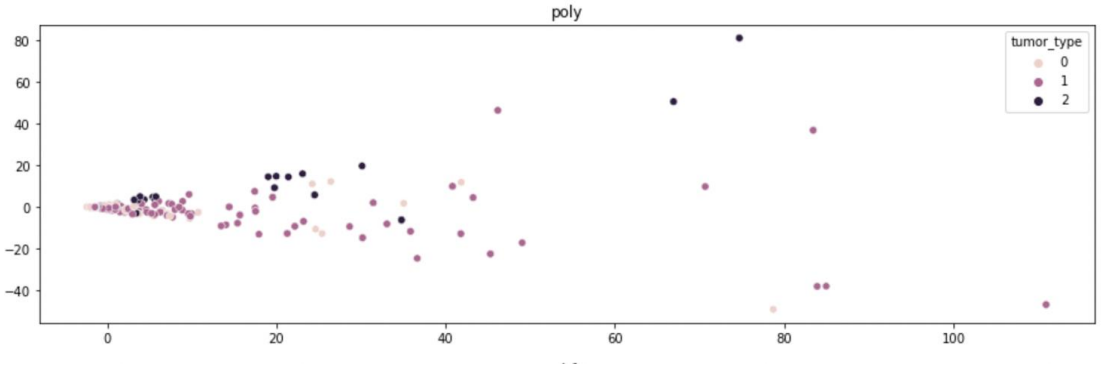
$$\Phi : \mathbf{R}^N \to F, \qquad \mathbf{x} \mapsto \mathbf{X}.$$

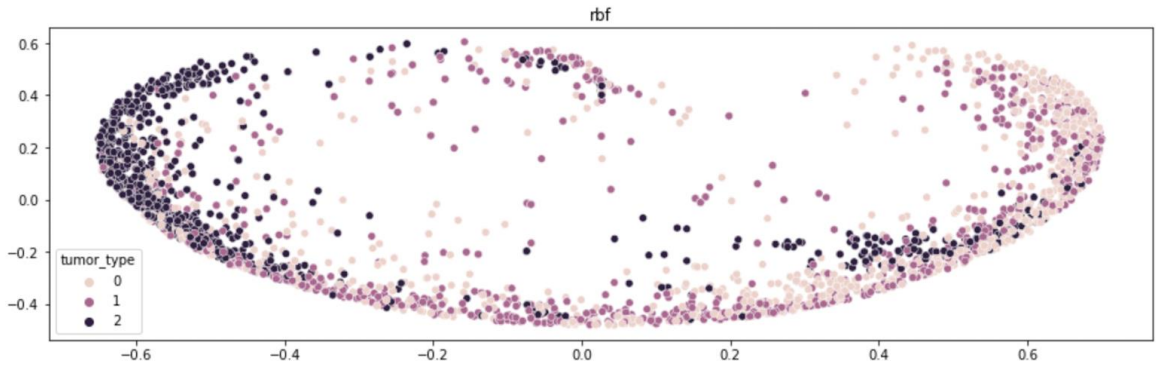
$$K_{ij} := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)),$$

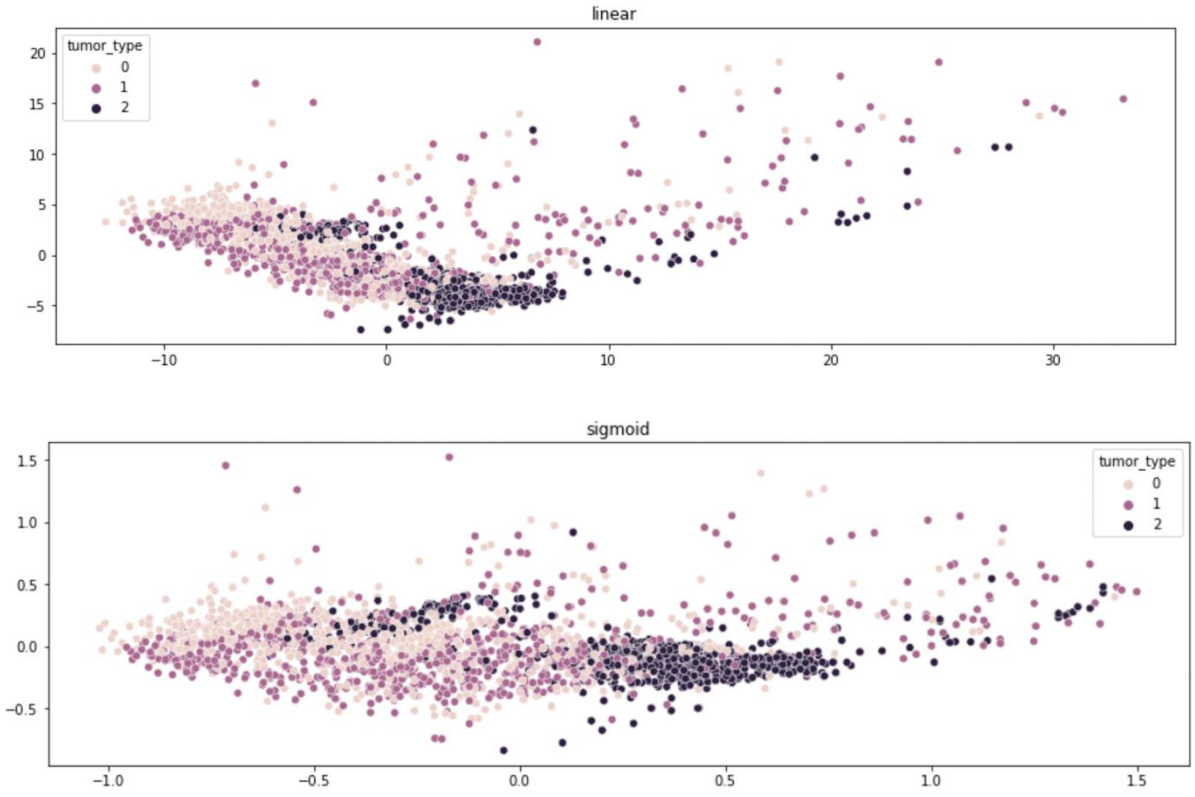
$$1 = \sum_{i,j=1}^{\ell} \alpha_i^k \alpha_j^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = (\boldsymbol{\alpha}^k \cdot K\boldsymbol{\alpha}^k) = \lambda_k (\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k).$$

We have compared how the data is transformed using all the four kernels provided -

1. Poly
2. rbf
3. Linear
4. Sigmoid







❖ Machine Learning Models

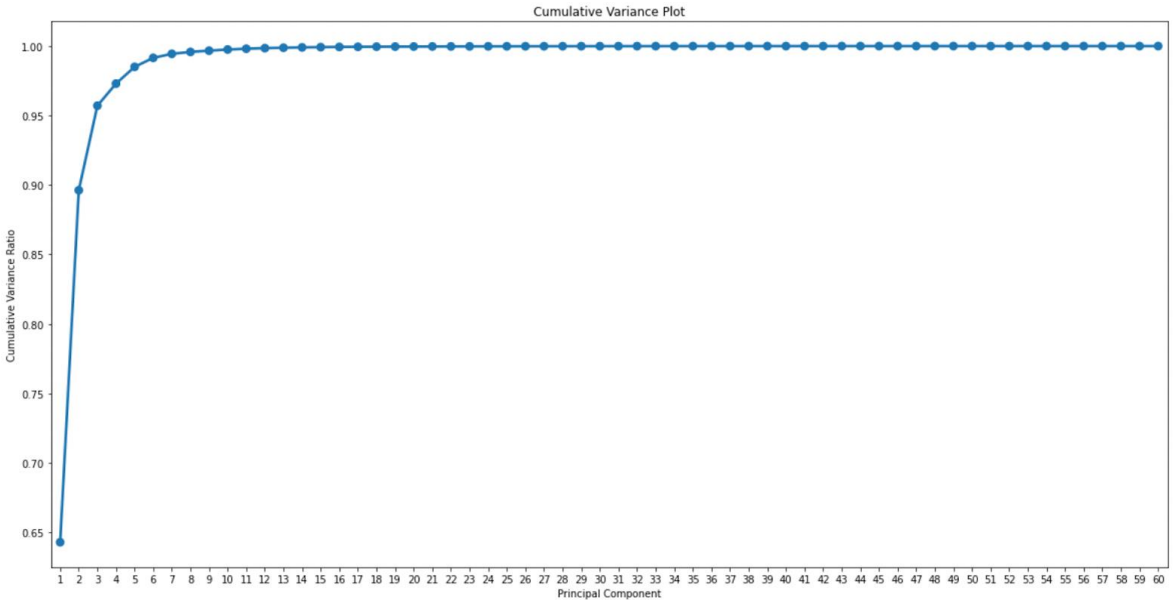
We have decided to use a set of ML models from each category for predicting the Brain Tumors after reducing the dimensionality. This also helps us understand how the different PCAs stand for in this type of dataset:

1. Linear Models & KNN
2. Trees - Decision Trees (used in AdaBoost)
3. Ensemble - Random Forests
4. Boosting - AdaBoost

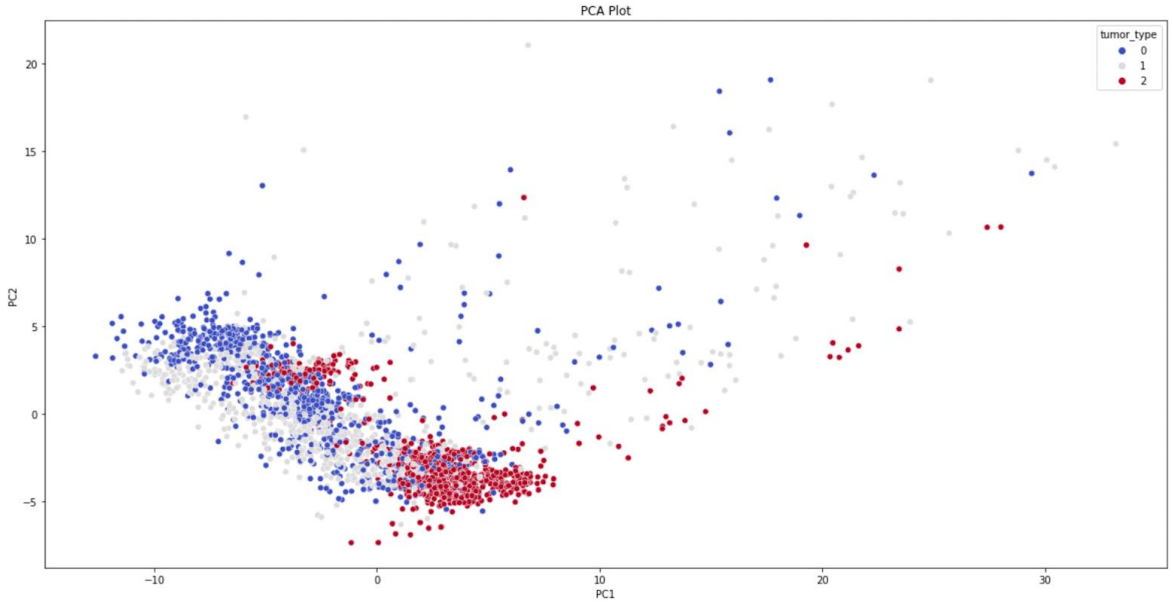
RESULTS

❖ PCA

Upon doing the **PCA**, we found that taking 6 components explained about 97% of the variance from the cumulative variance plot below:



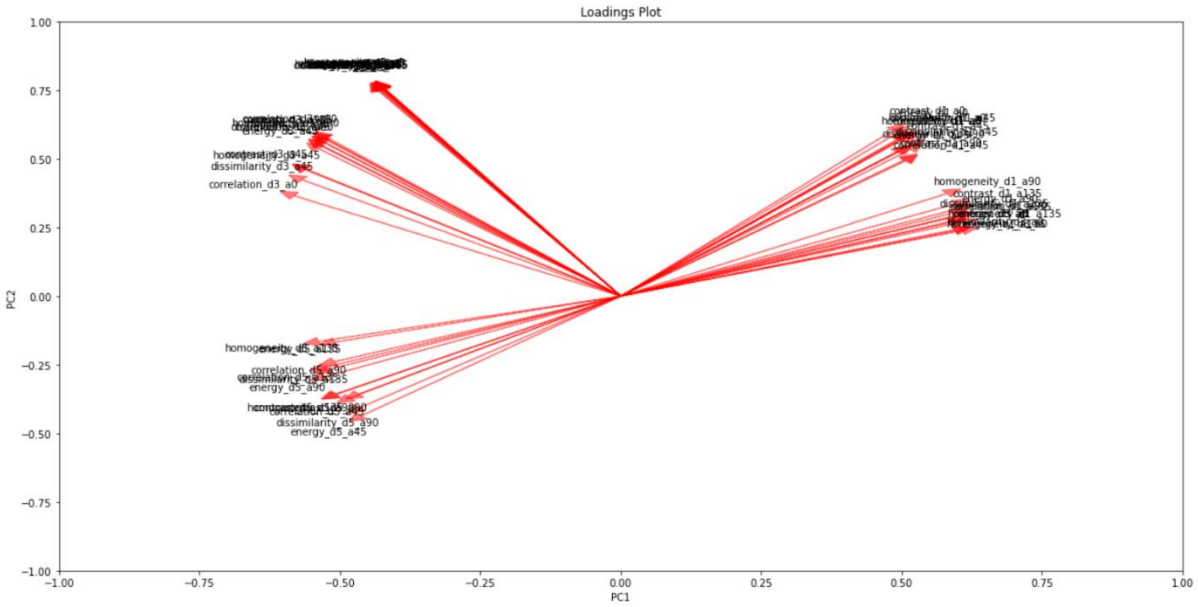
A PCA plot on the first two components helped us how the components were varied against the tumor types



As we can see the Tumor Type 2 is most likely easy to identify but the other tumor types are mixed.

Further, loading plots are a useful tool for understanding the relationship between the original variables and the principal components in a PCA analysis. They can help to identify

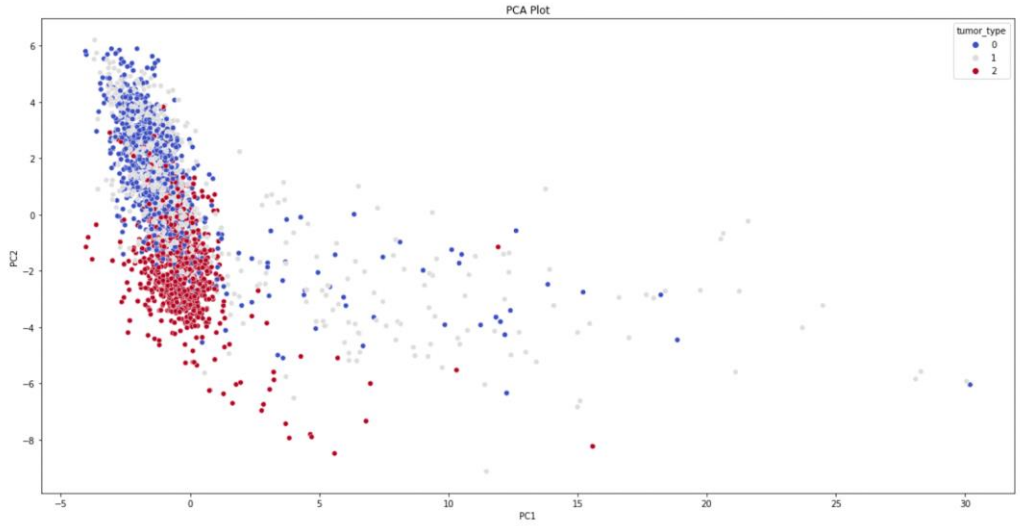
the most important variables in the dataset and to detect patterns and outliers that may be relevant for further analysis.



❖ Sparse PCA

We can clearly see the difference between Sparse PCA and the Standard PCA from the PCA plot of the first two components. We can clearly see that loadings have becomes sparse.

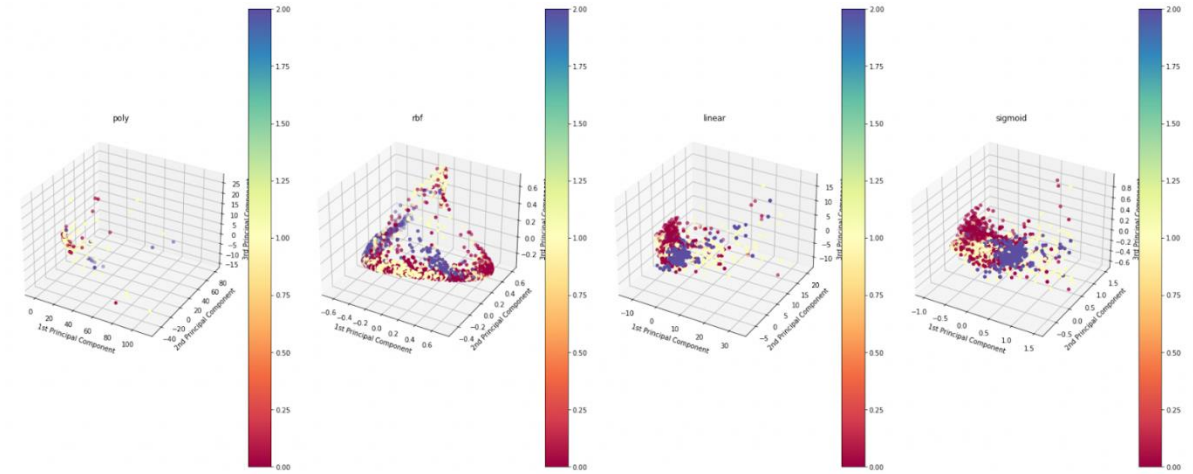
```
array([ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
       -1.27136657e-01, -1.45773836e-01, -1.48153872e-01, -1.41075045e-01,
       -1.51361598e-01, -1.60530046e-01, -1.62872353e-01, -1.54686246e-01,
       -1.56722104e-01, -1.54413388e-01, -1.54780656e-01, -1.51851254e-01,
        3.15333627e-01,  2.80485675e-01,  3.06612772e-01,  2.81756768e-01,
        2.35634940e-01,  2.24742410e-01,  2.25391800e-01,  2.26414890e-01,
        2.12649397e-01,  1.97600301e-01,  2.05629185e-01,  1.99177449e-01,
        1.30913905e-02,  3.55567764e-03,  1.25314150e-02,  3.49563097e-03,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  1.45127114e-04,  0.00000000e+00])
```



We calculated the Fraction of Variance Explained using the formula and found it to be 0.9758.

❖ Kernel PCA

For the **Kernel PCA**, we plotted the 3D maps for the first three components using all the kernels. We can clearly see that the data is not non linear therefore Kernel PCA might not be the best fit for this type of data.



❖ ML Modeling

Now, we move on to classifying the brain tumor after finding the components using different ML models.

For PCA, the initial accuracy scores of the linear ML models like LDA, QDA were low, however KNN got a better accuracy of 73% compared to the other models. The best model was Random Forests which achieved a score of 81%.

By Using K Fold Cross Validation, Random Forests on using K Fold Cross Validation gave a score of 82%. Upon using a GridSearchCV, Random Forests gave a mean test score of 79%.

```
In [65]: param_grid = {
             'n_estimators': [100, 200, 300, 400],
             'max_depth': [5, 10, 15, 20],
             'min_samples_split': [2, 5, 10, 14],
             'min_samples_leaf': [1, 2, 4, 6]
         }
         rfc = RandomForestClassifier(random_state=40)

         grid_search = GridSearchCV(rfc, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
         grid_search.fit(X_train, y_train)

         # Print the best hyperparameters and the corresponding mean test score
         print("Best hyperparameters: ", grid_search.best_params_)
         print("Best mean test score: ", grid_search.best_score_)

         Best hyperparameters:  {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
         Best mean test score:  0.7973223012143364
```

Even the Boosting Algorithm AdaBoost did not perform well as it gave an accuracy of 67% only.

In our scenario, the results of ML modeling on Sparse PCA were somewhat similar to the scores for standard PCA with very slight Improvements.

Only Random forests again gave the best performance with 83% on using K Fold Cross Validation improving on about 3% after using PCA.

The other ML Algorithms did not perform well like with the components of Standard PCA but however have a better accuracy compared to the Standard PCA where AdaBoost gave an accuracy of 69% compared to 66% in the Standard PCA.

As assumed earlier that since the data is not non-linear, Kernel PCA might not be the best choice for this type of data, the ML model accuracy scores proved our assumption right.

We got lesser accuracy for Random Forests in all the kernels except the Linear Kernel (because the linear kernel is similar to using PCA). Therefore, we felt to stop our predictions here as Kernel PCA is not the best choice.

```
Kernel poly
The Cross Validated Mean Error Score for RandomForestClassifier(random_state=42) : 0.72721060355898
Kernel rbf
The Cross Validated Mean Error Score for RandomForestClassifier(random_state=42) : 0.7456593809785884
Kernel linear
The Cross Validated Mean Error Score for RandomForestClassifier(random_state=42) : 0.8143990617711037
Kernel sigmoid
The Cross Validated Mean Error Score for RandomForestClassifier(random_state=42) : 0.7641075032104202
```

CONCLUSION

From the results of our analysis, we can see that Random Forests is the best performing ML algorithm for classifying brain tumors using both Standard PCA and Sparse PCA. Although the accuracy score for Standard PCA was slightly lower than that of Sparse PCA, the difference was not significant. This suggests that using Sparse PCA did not improve the performance of the ML models significantly, at least not for the chosen set of algorithms.

We also observed that linear models such as LDA and QDA did not perform well on this dataset. This may be due to the fact that the relationship between the variables and the classes is not linear. In contrast, Random Forests, a non-linear model, was able to achieve better accuracy scores. Boosting algorithm AdaBoost did not perform well in both Standard PCA and Sparse PCA, indicating that it may not be the best choice for this type of data.

Our analysis also showed that Kernel PCA was not the best choice for this dataset, as the data was not nonlinear. This is because Kernel PCA works best for datasets that exhibit nonlinear relationships between the variables. Thus, it is important to carefully choose the appropriate PCA method based on the type of data being analyzed.

It is also interesting to note that on using PCA, Sparse PCA Random Forests were able to get a better prediction performance on classifying Glioma and Pituitary tumors while it suffered to predict Meningiomas where using PCA Glioma had the best precision while Pituitary for Sparse PCA.

PCA:

	precision	recall	f1-score	support
0	0.84	0.76	0.80	292
1	0.78	0.79	0.79	286
2	0.83	0.91	0.87	252

Sparse PCA:

	precision	recall	f1-score	support
0	0.82	0.73	0.77	292
1	0.75	0.80	0.78	286
2	0.87	0.92	0.89	252

In conclusion, the results of this study show that using Random Forests with Standard PCA can be an effective approach for classifying brain tumors. However, it is important to note that the performance of the ML models may vary depending on the dataset and the specific algorithms used.

The dataset has been limited by the number of images in each tumor type and the image extracted by GrayCoMatrix might lose some of `the information from the images and these points might have affected our final results. It will be interesting to compare these results with Image Classification using Deep Learning and see how this approach stands. This can indeed be one of the future scope of the project. There are other methods of PCA like the incremental PCA which has not been explored as part of this project which can be extended later.

LITERATURE CITED

1. Sparse Principal Component Analysis by Hui ZOU, Trevor HASTIE, and Robert TIBSHIRANI
2. Kernel Principal Component Analysis
3. Brain Tumors
4. A Guide for Sparse PCA: Model Comparison and Applications
5. Sparse PCA Variance Explained
6. ChatGPT
7. Kaggle Brain Tumor MRI Dataset