Chapter 3

IDENTIFICATION OF NEURO-SEMANTIC REPRESENTATION OF CONCRETE NOUNS USING CASCADED FEATURE SELECTION

This chapter analyses the representation of concrete nouns in the brain. Words belonging to similar categories are represented in similar brain regions. It becomes hard to classify brain regions activated by similar expressions. This chapter proposes a cascaded feature selection technique to classify brain regions that are activated by concrete nouns.

3.1 INTRODUCTION

The depiction of words in the brain has been studied in numerous researches in the area of psychology, neuroscience, linguistics, and computer science, providing versatile theories about the representation of the meaning of words in the brain. Some of them propose that meanings are encoded in sensory-motor cortical areas, others claim that the gist of words has different representations based on semantic categories of words. The intangible sense of words and pictures stimulates different brain activity. It has been found that semantic common between different individuals is distributed continually across the cortex. [220] Carlson et al. (2014) [221] have used brain activation for correlation analysis in ventral temporal pathways. In contrast, Anderson et al. (2013) [222] have done correlation analysis of the whole brain and compared the HAL-based textual semantic model with BoVW visual semantic model but failed to predict differential collaboration of semantic models with brain regions. Fyshe et al. (2014) [223] have used brain activation data in textual word embedding in the JNNSE model, where word embedding was appraised as latent representation using matrix factorization to reconstruct the brain activation pattern. In [224], authors have adopted the SWE model [122]- a model developed from skip-gram to integrate brain activity data into word embedding assessment. Tom Mitchel et al. (2008) [225] have used multiple linear regression using 25 sensory verbs as an intermediate semantic feature to predict the fMRI activation in the brain for concrete nouns. Using multivariate multiple regression, Chang et al. (2010) [226] proposed a generative classifier that models the hidden factor that fortifies the semantic picture of object knowledge. They used k-NN and SVM classifiers to decode the word linked with brain activation. The authors [227] specify that individuals have perceptions in common, but their brain coding is alike; however, the location and activation levels may not be the same across people. The research founds a new means of recitation brain activity using factor analysis and ML-generative model-GNB and linear regression to fMRI data for the estimation. In [228], authors have

selected to exercise the mean activation of 116 areas of interest as constrained by the Automated Anatomical Labelling (AAL) atlas rather than examining the raw voxel statistics existing in the dataset. ROI activations are used as input to the classifier, enabling understanding since outcomes can be reviewed at the ROI level instead of at the single voxel level. The most crucial influence is getting comprehensive group consequences with joined data from all nine subjects. The model can predict the fMRI activation for words beyond the training set. The authors in [229] discovered that dependency parse-based structures are the most real, and the arrangement performance is better for a corpus-based model. The simple semantic features with steering information provide a suboptimal solution at a much lesser computational cost. All these methods prove that semantic features of the word are associated with neural stimulation related to the thinking word (Mitchell et al., 2008) [225].

3.2 METHODS AND MODELS

3.2.1 DATASET DESCRIPTION

We have used a dataset available online [225]. The data was collected from 11 right-handed persons of whom 5 were female. Each participant was shown a combination of 60 concrete noun- picture (line drawing) pairs on the screen and fMRI recording for each such stimulus was captured. The words shown were of 12 semantic categories and 5 examples from each category as shown in the table. Each word-picture pair was shown 6 times randomly hence a total of 360 fMRI images were collected for each participant. The stimulus was shown for 3 sec followed by 7sec fixation. The participants were asked to think of the properties of the noun-picture pair which they were watching on the screen.

A single image was created by taking the mean of the images collected at 4,5,6 and 7 sec after the onset of the noun and its line drawing to compensate for the

hemodynamic latency. The data acquisition was done on Siemens Allegra 3.0 T scanner. The experiment was performed with a gradient echo, EPI sequence with TR=1000ms, TE=30ms and flip angle=600. Seventeen 5 mm thick oblique axial slices were imaged with a gap of 1 mm between the slices. The acquisition matrix was 64*64 with 3.125mm*3.125mm*5mm voxels. Processing of the dataset was done with SPM2. The data was corrected for slice timing, motion and was spatially normalised into MNI space and resampled to 3*3*6 mm3 voxels.

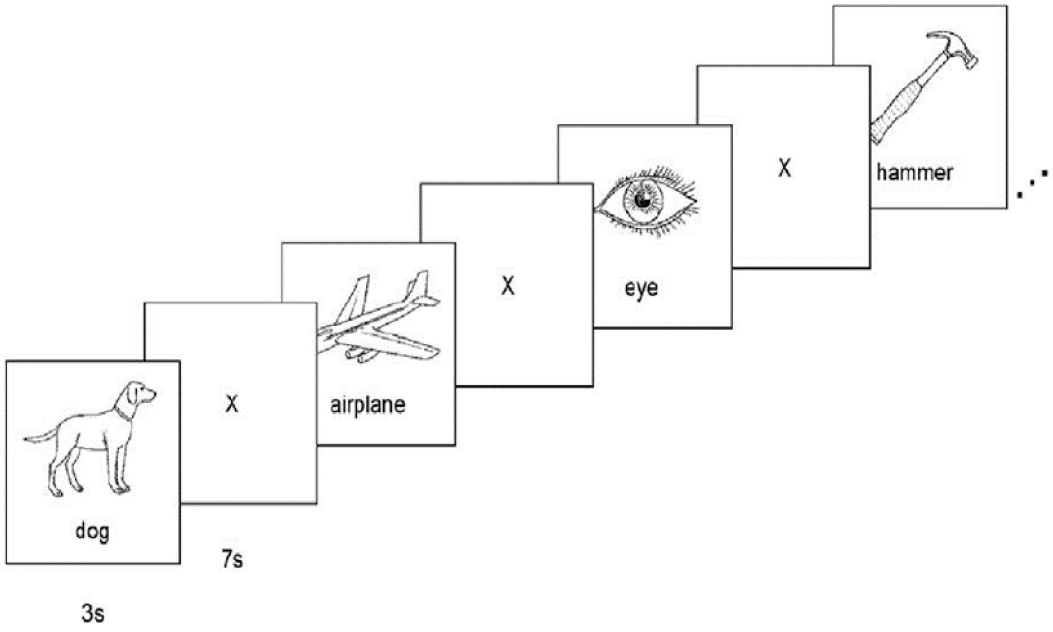

FIGURE 3.1: Data acquisition scheme

3.2.2 MODEL DESCRIPTION

To classify the fMRI scans, we take two approaches, i.e., categorise them into two classes (living and non-living) or organise them into 12 different types. We start by normalising our dataset using SPM12 in MATLAB in both cases. After standardising our data, we apply feature selection techniques to reduce the dimensionality of our dataset. We first apply the variance threshold to remove the garbage values introduced in our dataset while normalising and then apply either PCA or LDA or

Table 3.1: Concrete nouns and its categories

Category	Exemplar 1	Exemplar 2	Exemplar 3	Exemplar 4	Exemplar 5
Body parts	Leg	Arm	Eye	Foot	Hand
Furniture	Chair	Table	Bed	Desk	Dresser
Vehicles	Car	Airplane	Train	Truck	Bicycle
Animal	Horse	Dog	Bear	Cow	Cat
Kitchen Utensils	Glass	Knife	Bottle	Cup	Spoon
Tools	Chisel	Hammer	Screwdriver	Pliers	Saw
Buildings	Apartment	Barn	House	Church	Igloo
Building Parts	Window	Door	Chimney	Closet	Arch
Clothing	Coat	Dress	Shirt	Skirt	Pants
Insect	Fly	Ant	Bee	Butterfly	Beetle
Vegetable	Lettuce	Tomato	Carrot	Corn	Celery
Man-made objects	Refrigerator	Key	Telephone	Watch	Bell

a combination of both to remove the correlation within the features of our dataset. Finally, we classify the data using three techniques, i.e., Naive Bayes, Random Forest, and Multi-Layer Perceptron. We used ten-fold cross variation and tuned the hyper-parameters on a case-by-case basis. The machine learning methods used in this work requires 3D images to be converted into discrete format; for this each 3D images were split into 21768 voxels, each of size 3mm*3mm*6mm and having its own activation value.

SPATIAL REALIGNMENT AND RELOCATION: All the fMRI scans are not oriented in the same direction or are aligned in reference to some standard location. Before we relocate the voxel data in a standard space, we need to make sure that the voxel structure of the brain for all the objects is oriented in the same direction. We use the Standard rigid body transformation technique to reorient and realign all the scans.

Every person has a unique brain, both in terms of volume and structure. The size of the brain and the neural network of the brain of any two people are different. This is not ideal for feature extraction. If we take two subjects and realign their brain in the same direction and localise the centroid of their cerebellum to origin, even then the voxel corresponding to the same coordinates of the brain of the two subjects may

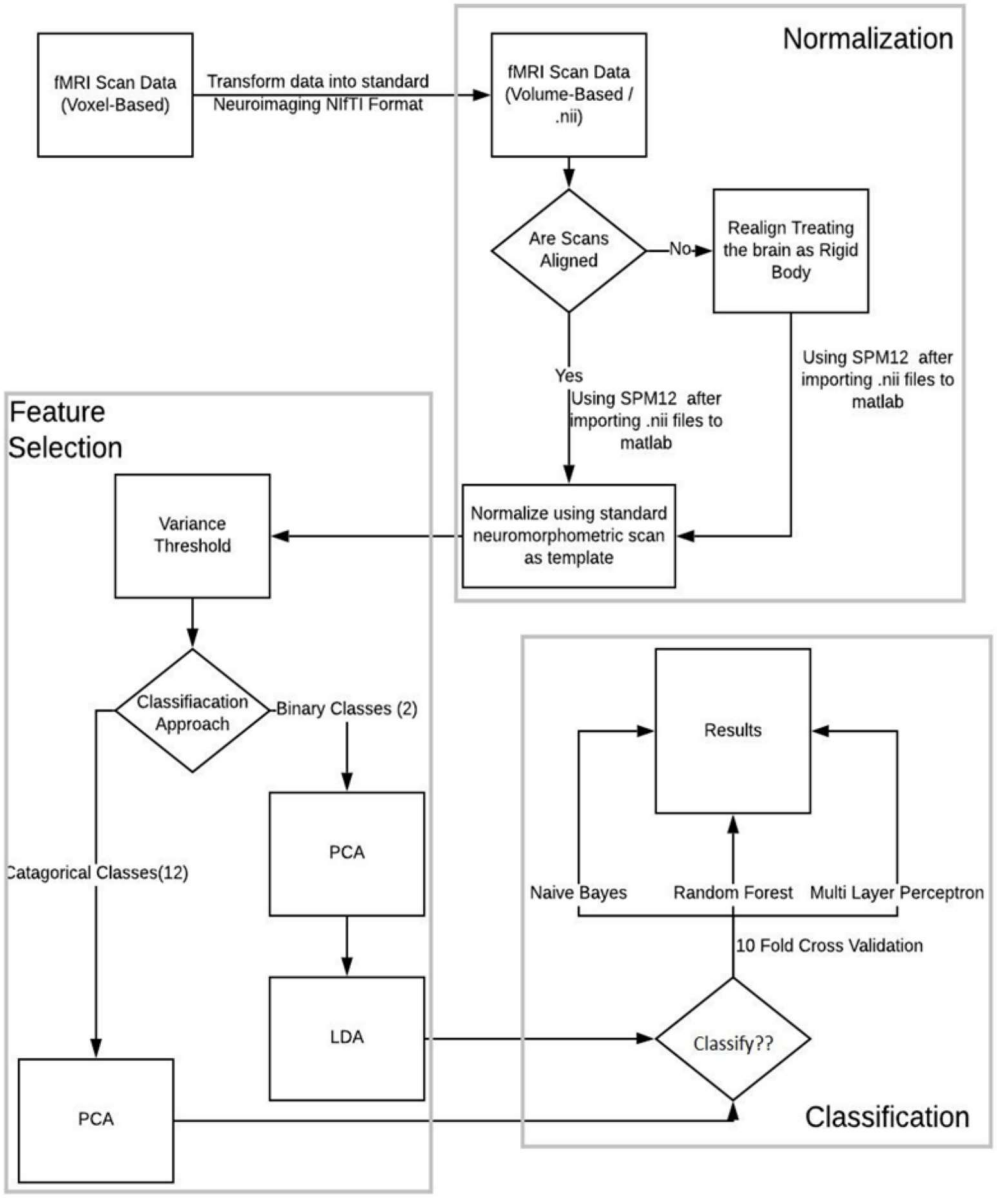

FIGURE 3.2: Proposed Model

belong to different regions of the brain, both responsible for two completely different activity. Because of this inconsistency in 'location' of voxel and the 'function' of the region of the voxel amongst different subjects, we get good results when training with data of one person and terrible results when training with data of various subjects.

NORMALISATION: For Normalisation, we used a standard Neuro-morphometrics

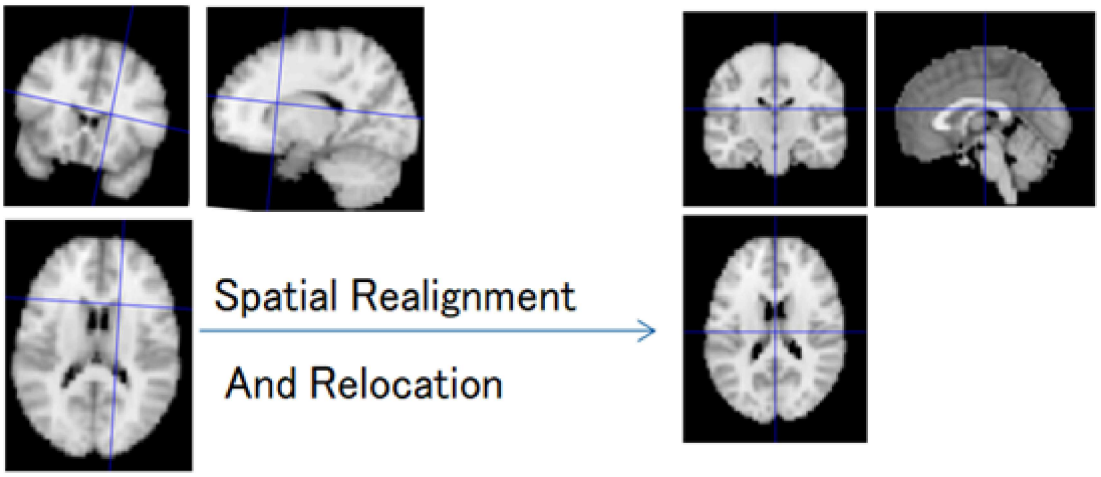

FIGURE 3.3: Spatial Realignment

template that comes with SPM12. All the various regions of the brain are highlighted with different intensity levels that have been averaged from a large testing corpus. Since our template used highlighted intensity levels, we do not only preserve the structure of the template but also preserve the functional structure of the brain, as voxel data of one region of the brain is warped into its functional region only. Since the intensity level of all the regions has been averaged from a large testing corpus, the amount of noise introduced is minimum.

WARPING ONTO STANDARD TEMPLATE: In the previous step, where we use rigid body transformation matrix, here we use Affine Transformation Matrix, therefore, although we are warping various regions of the brain onto fixed functional regions, we are still maintaining the overall structure of the brain. These functional regions are the same for all the subjects, therefore we no longer have an inconsistency in the 'location' of voxel and the 'function' of the region of that voxel amongst different subjects.

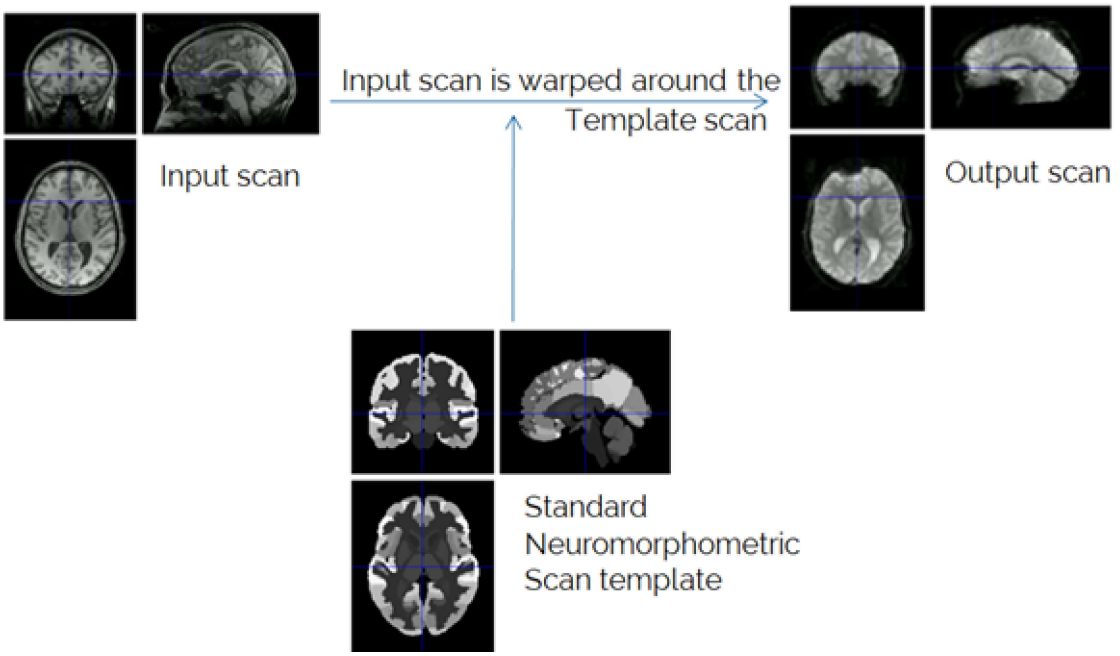

FIGURE 3.4: Normalisation

$$\text{Transformation Matrix} = \begin{pmatrix} r_{xx} & r_{xy} & r_{xy} & t_x \\ r_{yx} & r_{yy} & r_{yx} & t_y \\ r_{zx} & r_{zy} & r_{zx} & t_{zx} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The goal of this stage is to make sure that by the end, in the brain structure made by the voxel data, the frontal lobe is facing y direction, the cerebellum is facing -z direction and that the centroid of interventricular foramen is at origin. Not all voxels of the fMRI image are equally important. Before we normalise the data, we had the data in the form of voxels, i.e., we had information about the coordinate of the voxel along with the intensity of that voxel. The no of voxels constituting the brain of each subject ranged from 19000 to 21000. However, after normalisation, we get a 3D array of size [54x64x27], i.e., more than 90000 features describing each fMRI scan.

VARIANCE THRESHOLD: Variance threshold is the most basic and rudimentary approach for feature selection for fMRI scans. It is one of the most popular

feature selection methods; it pulls out all the features whose variance doesn't pass some threshold. By default, it detaches those features whose variance is zero, i.e., features which have the same values in every sample. The researchers assume that the features that have more variance are useful and consist of essential information. An important thing to note is that this technique is not considered the relationship between the feature or feature correction over the target label.

Algorithm 1 Feature Selection:- Variance Threshold

InputInput OutputOutput

Normalized dataset Dataset without NaN or quasi constant features **begin**

for each voxel, v in the image

Find the mean of that voxel for all the scans

Find the Variance of voxel v

if variance of voxel v is more than V_{th}, keep it, else discard it, where V_{th} is the pre-defined threshold for Variance in our algorithm

end

For every voxel at a coordinate, suppose x,y,z, if the intensity of the voxel is barely changing for all the scans, we can assume that it is not that important in determining the classification result. If we know that regions of the brain are highly correlated, thus we can rest assured that no vital information is permanently lost. Most of the additional values that were added during normalisation were trash values; they were either Nan or zeros. We first turn all Nan values to zero. Using the variance threshold, we remove all these trash values and reduce the number of features drastically so that all other Feature Selection methods can be applied efficiently.

p = probability Voxel x has zero intensity

q = Probability Voxel x has non-zero intensity

$$Var[x] = p * q \tag{3.1}$$

For our model, we chose 'p' as 0.9; therefore, if more than 90 percent of scans had the intensity for some coordinate as zero, then it would be removed. For our model, if Var[x] were less than 0.09, we would discard that Voxel.

QUASI-CONSTANT FEATURES- These are the almost constant features. In other words, these features have the same values for a substantial subset of the outputs. Such features are not very useful for making predictions. There is no rule as to what should be the threshold for the variance of quasi-constant features. However, as a rule of thumb, remove those quasi-constant features that have more than 99% similar values for the output observations. Feature selection techniques are employed to remove quasi-constant features from the dataset as these features have no role in determining model accuracy.

PRINCIPAL COMPONENT ANALYSIS: It is a dimensional reduction technique mainly used when the dataset has many features and is extensive. It transforms more big data into a different variable of small chunks without significant information loss. Reducing the number of features from the data can compromise accuracy. The primary stump of PCA is to reduce the data variable without harming the information. Principal components are a collection of different points in real co-ordinate of the unit vector. The best fit line minimizes the mean squared distance from the line. So PCA is the procedure to compute the components and utilize them to change the features of data. There are mainly three steps involved in PCA, (i) Standardization, (ii) Covariance matrix computation, and (iii) computation of eigenvector and eigenvalues of the covariance matrix to detect the principal component. Principal component analysis (PCA) is an unsupervised technique used to pre-process and reduce the dimensionality of high-dimensional datasets while preserving the original structure and relationships inherent to the original dataset so that classification models can be trained on the reduced features and still make accurate predictions.

To standardize the data, we define the same range of data for every feature, the values of all instances for an attribute must lie in that defined range. We also use this to remove any outliers. However, since we have already normalized our data to a region-based standard fMRI brain scan template, we are unlikely to find many outliers. Also, because we performed a Variance threshold on our data, the no of variables to compute has been significantly reduced. We calculate a vector of the mean of all the features, and using that vector; we construct the scatter matrix. We then make a covariance matrix, computing the correlation between any two brain voxels. The covariance matrix is symmetric. A negative covariance value denotes that the two variables are indirectly proportional, and a positive value means that they are directly proportional to each other. A higher magnitude means that the correlation between the two variables is stronger. We then compute the eigenvalues using scatter and covariance matrix. The first eigenvalue is used to make the first Principal Component and has the most variance. As more eigenvalues are calculated, the variance in their subsequent Principal Components reduces.

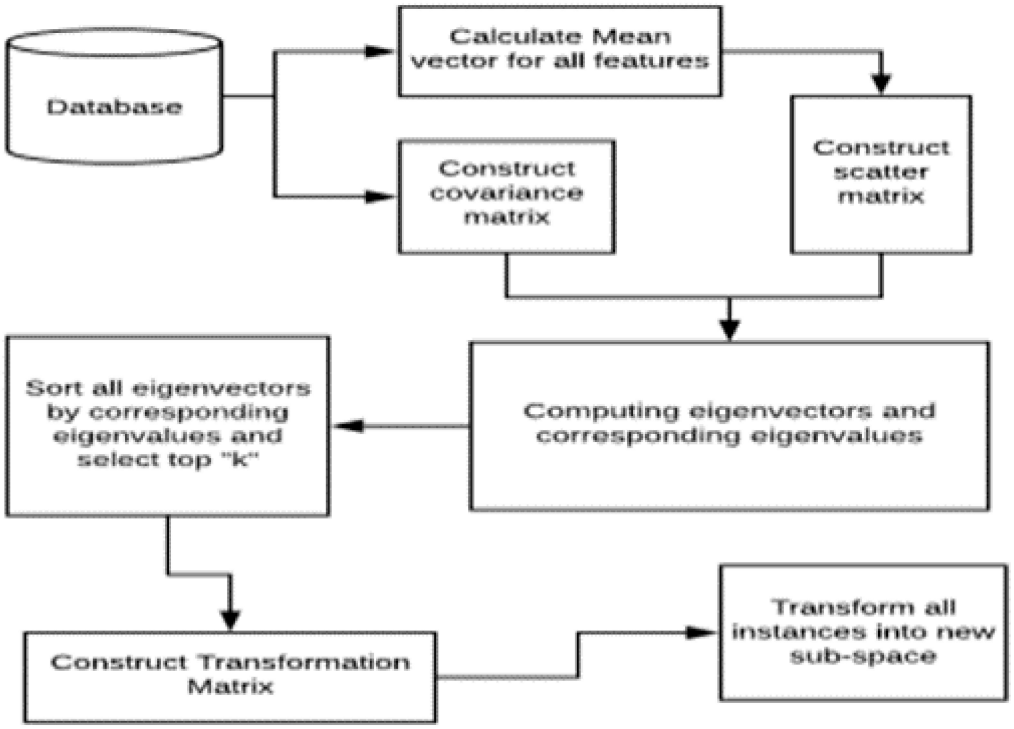

FIGURE 3.5: Principal Component Analysis

Algorithm 2 Feature Selection:- Principle Components Analysis

Input:Dataset

Output:Dataset with correlated features removed

1.begin

1.Calculate the mean vector, M, containing the mean of all the features

2.For a feature i, computer the matrix: $D_i = (X_i$ - M$)(X_i$ - M$)^T$

3.Repeat step 3 for every feature and then compute the Scatter matrix, S $S = \sum D_i$

4.For any two features, (X,Y), calculate their covarience, $Cov(x, y) = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{N}$

5.Repeat step 5 for every pairs of features and then construct the covarience matrix.

6.Using np.linalg.eig Computing eigenvectors and their corresponding eigenvalues using covariance OR scatter matrix.

7.Sort eigenvectors by their corresponding eigenvalues

8.Use the top k ranked eigenvectors to construct a tranformation matrix.

9.For every instance, compute the dot product of that instance with the transformation matrix

end

LINEAR DISCRIMINANT ANALYSIS (LDA): is similar to PCA by a linear combination of different feature vectors. It is mainly a supervised dimensional reduction method. It is closely related to the analysis of variance and regression analysis. These also represent the dependent variables as a linear combination of various features. Therefore, we cannot test the data we used to train the feature selection model. Our goal here is to minimize the variance between points in the same class (within-class variance) and maximize the distance between points of two classes (between-class variance).

Algorithm 3 Feature Selection:- Linear Discriminant Analysis

InputInput OutputOutput

Dataset Dataset with correlated features removed **begin**

For one class, calculate the mean vactor for all instances in that class

Repeat step 2 for every class

Construct the within class and between class scatter matrix.

Using np.linalg.eig Computing eigenvectors and their corresponding eigenvalues using the dot product of within class scatter matrix and the between class scatter matrix

Sort eigenvectors by their corresponding eigenvalues

Use the top k ranked eigenvectors to construct a tranformation matrix.

For every instance, compute the dot product of that instance with the transformation matrix

Like in PCA, we calculate the mean vector to find the scatter and covariance matrix. However, after seeing the eigenvectors, we train a supervised model to transform our data into a subspace in which the within-class variance is minimized, and between-class variance is maximized. We create a lower-dimensional projection space by maximizing the between-class variance and minimize the within-class variance. We used Native Bayes as a baseline for comparison. All the testing was done in Weka. We used two popular and reliable classification techniques to classify our data. It

also reached the classification of different classes; it mainly focuses on minimizing separability between known classes by generating a new axis (linear) and projecting those data points into that axis. It doesn't find the principal component; it mainly looks for the different features and data points in the subspace, which are more discriminated from the original data. It is more helpful when measuring an independent variable for every observation of continuous quantities.

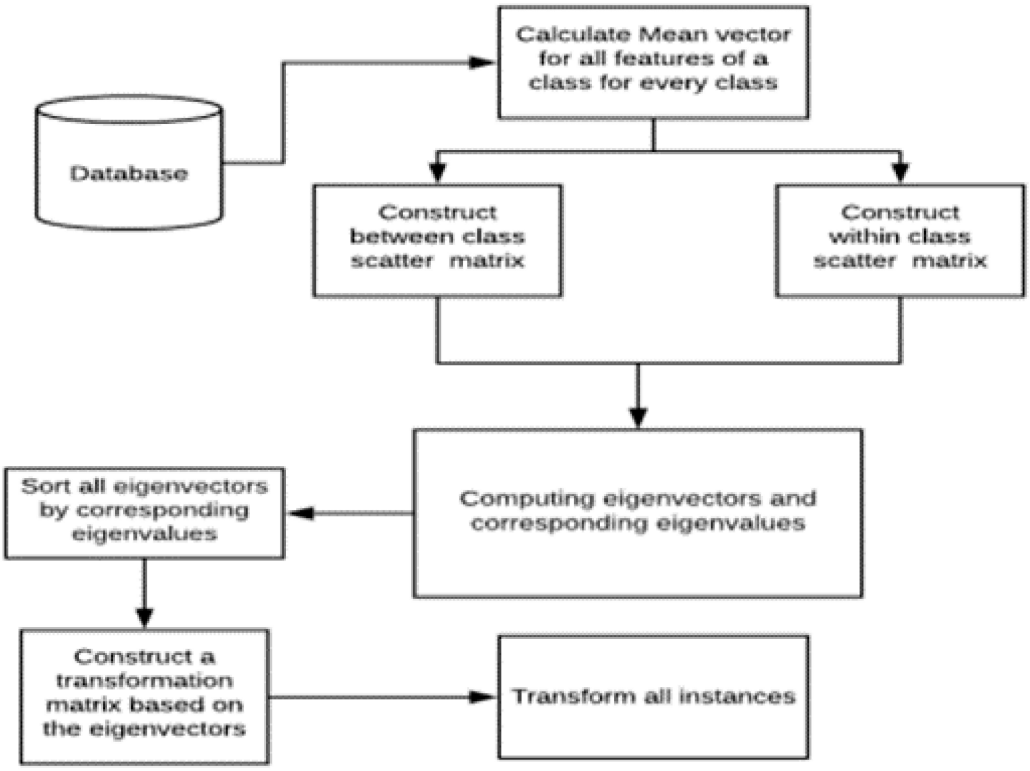

FIGURE 3.6: LDA

3.3 RESULT ANALYSIS AND DISCUSSIONS

In this chapter, we will be comparing the results of all our various pre-processing techniques with other traditional pre-processing techniques, some of which are currently used most commonly for best results. All the pre-processing of the data

was done in python using sklearn. Raw data was turned to standard .nii format for normalisation using nibabel in python. For normalisation, all FMRI scans were normalised using the SPM12 framework in MATLAB. Training for all supervised and unsupervised feature selection was done on data of 8 subjects (unless another approach to training and testing is specifically defined before a result). The data of the last subject was transformed using those trained models. Finally, classification was performed on the data of the last subject. This was done to remove most of the bias from our model. All training and testing for classification was done, and all hyper-parameters were tuned on a case-by-case basis. Additionally, 10-fold cross validation was used during classification to get a better estimation of out models bias and variance in unknown test cases.

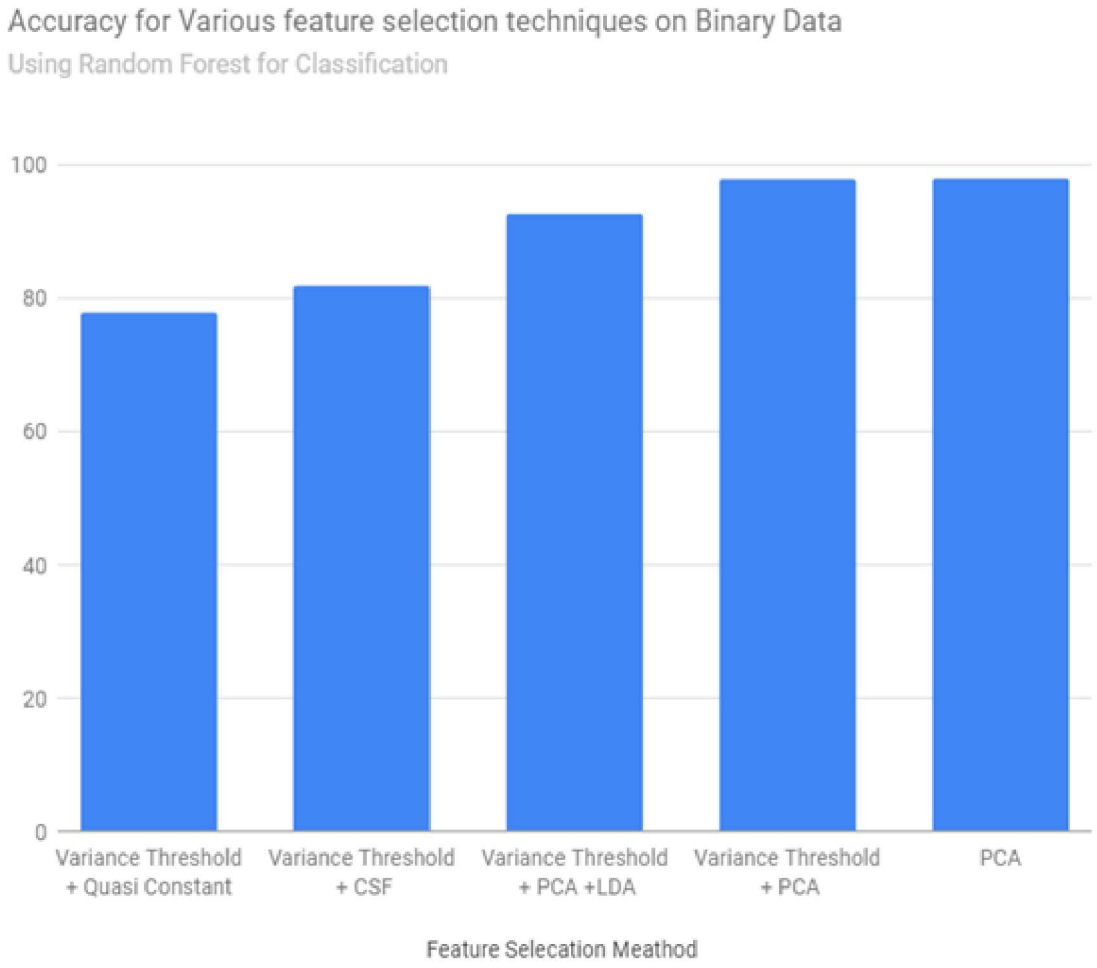

FIGURE 3.7: Feature selection Accuracy Result

Figure 3.7 shows the accuracy of various combinations of pre-processing that can be applied on the data. For reliable comparison, we used random forest for classification. For binary classification, the data was highly skewed (with a ratio of 9:2), hence weighted costs were used for classification. Because there were only two classes to classify in, we got 78.0% accuracy using just Variance Threshold and Quasi Constant. However, after using PCA, we got 98.49% accuracy. It should be noted that using PCA without variance threshold, we got 98.91% accuracy, however the training time of the model increased by a factor of around 27. By using LDA on top of PCA, our accuracy decreased to 92.80% however, we were able to reduce the testing time by a factor of 10.

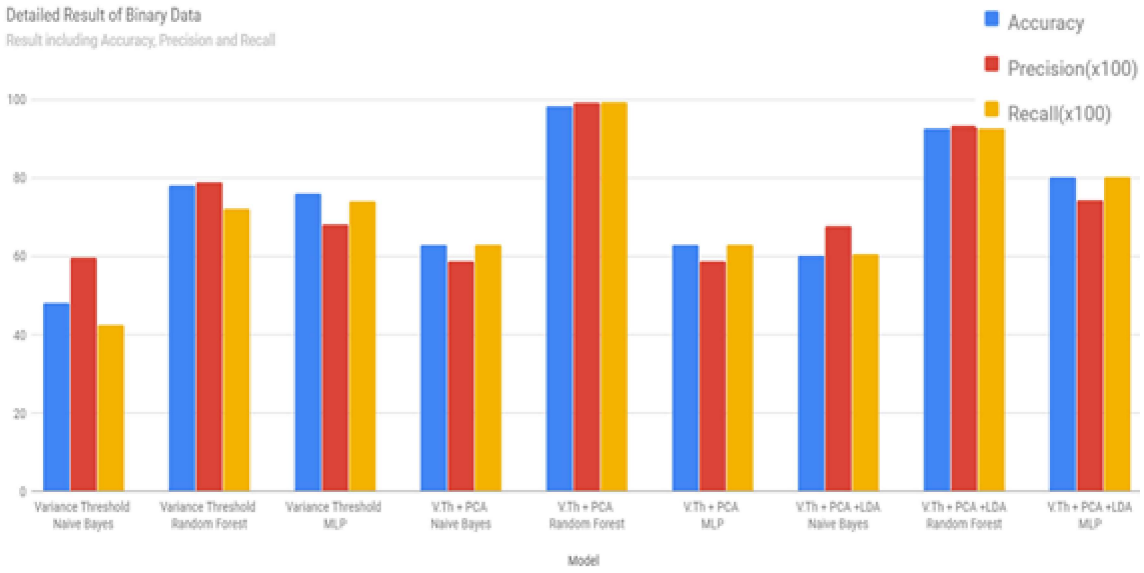

FIGURE 3.8: Classification Result

Fig.3.8 shows the accuracy, precision and recall for three pre-processing models using various classification techniques. In each case, Naive Bayes acts as a baseline for judging the capability of the pre-processing technique. Going from only variance threshold to variance threshold followed by PCA followed by LDA, we saw an improvement of 11.89%, 14.02% and 4.32% in Naive Bayes, Random Forest and Multilayer Perceptron respectively. If we compare variance threshold followed by PCA followed by LDA with variance threshold followed by PCA, we saw that for the simpler model performs better by 3.72%, 5.85% and 7.76% in Naive Bayes, Random

Forest and Multilayer Perceptron respectively. Since we applied a cost sensitive learning approach while training our models, we see that precision and recall for all our models aren't skewed. In fig 3.9. we can see that for categorical data, even using

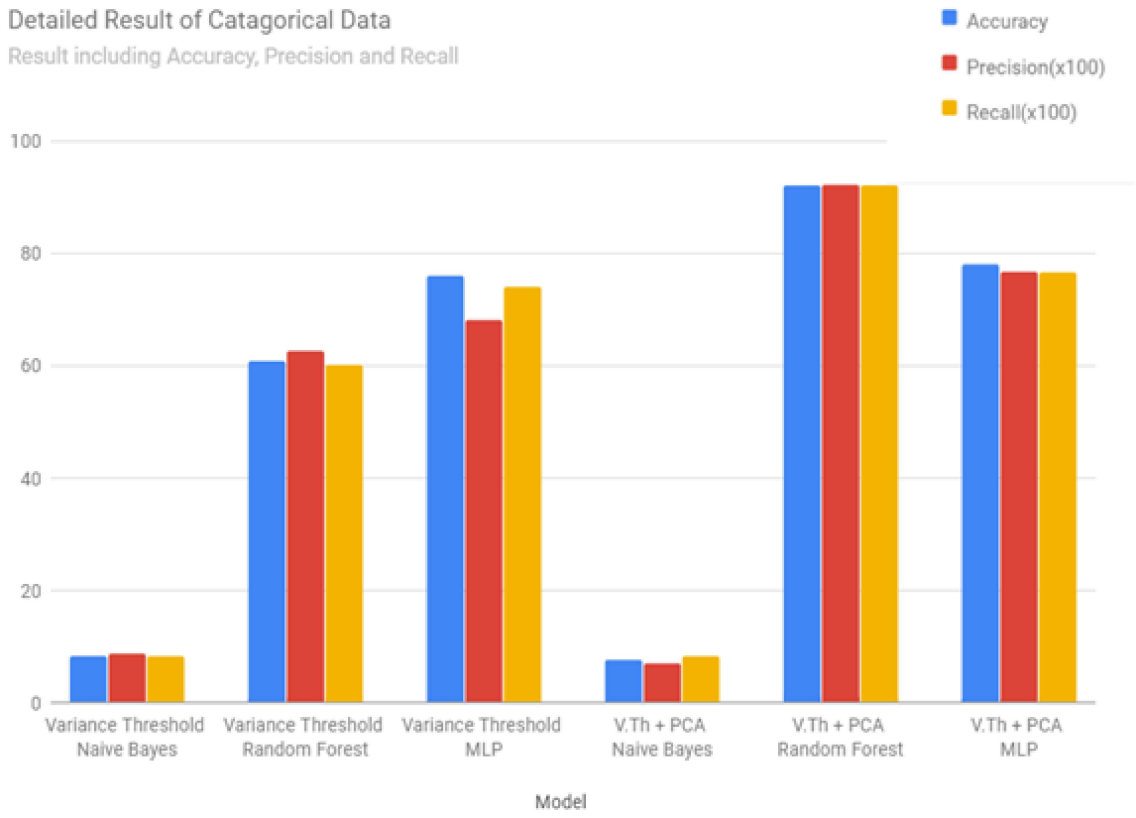

FIGURE 3.9: Result for 12 Class problem

conventional methods like cfs to remove correlation gives us overall bad results. Our simple approach of Variance Threshold although takes longer to train, but gives overall better results. In Figure 3.9 we can see that because of the high number of classes, in all cases, Naive Bayes isn't capable of classifying the data accurately. We got our best result using Variance threshold followed by PCA with an accuracy of 83.51%. We improved our accuracy by 24.67% over our base model of just Variance threshold. Table shows the confusion matrix for a model trained on a random forest classifier using Variance threshold followed by PCA with 10-fold cross validation.

Table 3.2 shows the confusion matrix for a model trained on a random forest classifier using Variance threshold followed by PCA followed by LDA with 10-fold cross

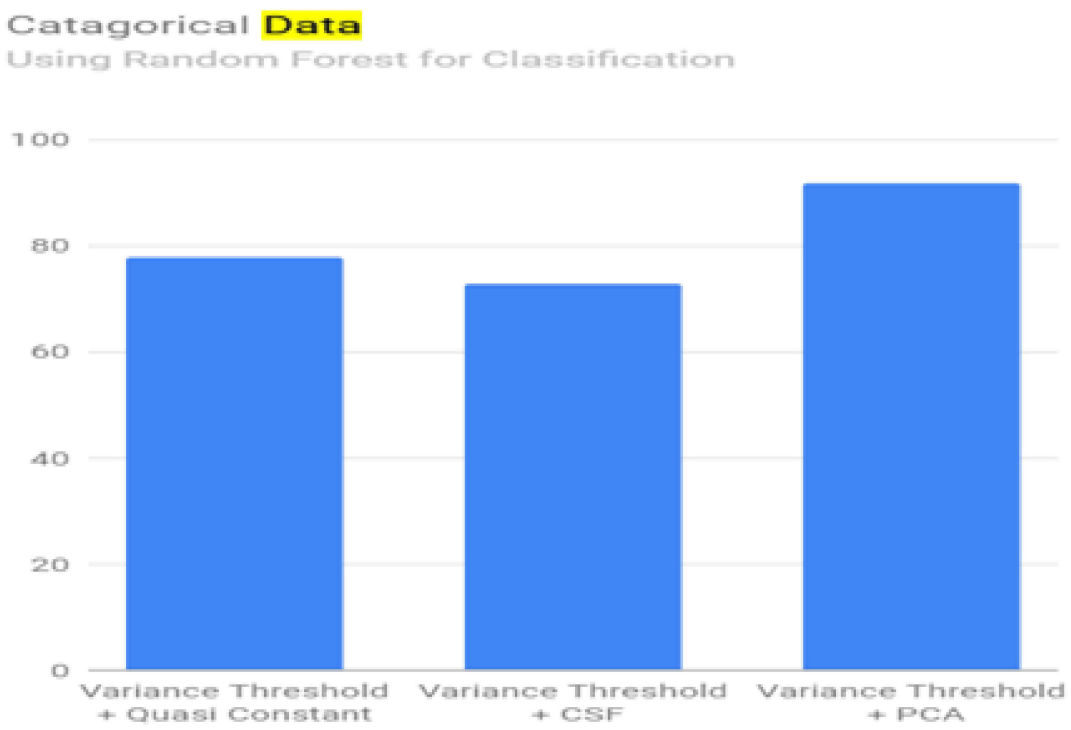

FIGURE 3.10: Efficiency of cascaded feature selection

validation. For this model, we did not transform a testing set on our feature selection model for training and testing our classifier. We directly used the transformed training data from our PCA and LDA model and used that to train and test the random forest classifier with 10-fold cross validation. Since LDA is a supervised learning technique, the transformed data we got from the training model was structured to be able to classify effectively for these cases only, hence we see some form of over-fitting if we study the confusion matrix in table 3.2.

In this study, we were able to classify with an accuracy rate of 98.49% if the subject was thinking about a living thing or a non-living object. Our proposed model predicted these results for subjects with no scans used to train the model. Using the same principle for finding results, we got an 83.51% accuracy when instead of "living and non-living", we divided all objects into 12 categories, as proposed by Tom M. Mitchell [225].

If we study the confusion matrix in table 3.3, we can see that even if we train the

Table 3.2: **Confusion Matrix**

A	B	C	D	E	F	G	H	I	J	K	L	Category
217	7	3	1	7	5	6	4	4	7	4	5	A=Body parts
4	233	3	5	2	8	3	2	2	4	0	4	B=Furniture
1	5	237	3	3	3	2	3	2	3	4	4	C=Vehicles
2	4	4	234	2	1	3	7	5	2	2	4	D=Animal
7	3	1	5	225	7	3	2	3	3	7	4	E=Kitchen Utensils
5	5	4	8	1	226	3	6	4	1	4	3	F=Tools
3	4	3	6	3	5	221	2	7	5	5	6	G=Buildings
4	3	9	7	7	6	4	220	4	1	3	2	H=Building Parts
4	2	4	5	7	3	3	3	225	3	4	7	I=Clothing
6	5	6	7	4	2	5	4	5	216	5	5	J=Insect
1	2	3	3	1	5	5	2	7	3	237	1	K=Vegetable
3	7	4	4	2	3	8	8	3	6	7	215	L=Man-made objects

Table 3.3: Confusion Matrix 12 Class

A	B	C	D	E	F	G	H	I	J	K	L	Category
237	0	0	0	0	0	0	0	33	0	0	0	A=Body parts
0	140	0	0	48	0	0	4	0	78	0	0	B=Furniture
0	0	270	0	0	0	0	0	0	0	0	0	C=Vehicles
0	0	0	187	0	0	42	0	0	0	41	0	D=Animal
0	117	0	0	119	0	0	19	0	15	0	0	E=Kitchen Utensils
0	0	0	0	0	270	0	0	0	0	0	0	F=Tools
0	0	0	2	0	0	239	29	0	0	0	0	G=Buildings
0	3	0	0	26	0	2	239	0	0	0	0	H=Building Parts
74	0	0	0	0	0	0	0	150	0	0	46	I=Clothing
0	52	0	0	6	0	0	0	0	212	0	0	J=Insect
0	0	0	34	0	0	0	0	0	0	232	4	K=Vegetable
18	0	0	0	0	0	0	0	104	0	6	142	L=Man-made objects

model on data that has been transformed to specifically give good results on that data, we see that for every class, it at most wrongly classifies the instance into two or three classes only. Upon further examination, we can see that in almost all cases, if one class is wrongly classified into some other class then the latter class is also wrongly classified into the former class. From this we can infer that not all classes are equally independent of each other, for example objects of class 'mammal' and 'insect' should clearly be a lot similar to each other than they are to objects from class 'furniture' or 'tool'.

3.4 CONCLUSIONS

This chapter proposes a cascaded feature selection technique to classify the brain signal for concrete nouns. Similar words activate the similar regions of the brain hence it becomes very hard to differentiate between the brain signals for similar categories of words. Employing cascaded feature selection, we become able to classify the brain regions for different concrete nouns. In this study we have classified the brain activation of concrete nouns in binary classes with accuracy of 98.49% whereas in 12 classes our model could classify with accuracy of 83.51%. In the 12-class problem when without using cascaded feature selection we tried to classify the nouns the result was of 12% accuracy but with cascaded feature selection result enhanced to acceptable limit.