Xometry data challenge report

Dataset Description

The dataset provided for the challenges is a masked multivariate dataset with independent variables from x0- x11 and a dependant variable y

Independent features

x2,x7,x11 - Binary variables

x0 - x11 (except features x2,x7,x11) - Numeric variables

Dependant features

y - continuous numeric variable

From the description it was clear that the dataset had multiple features.

Objective

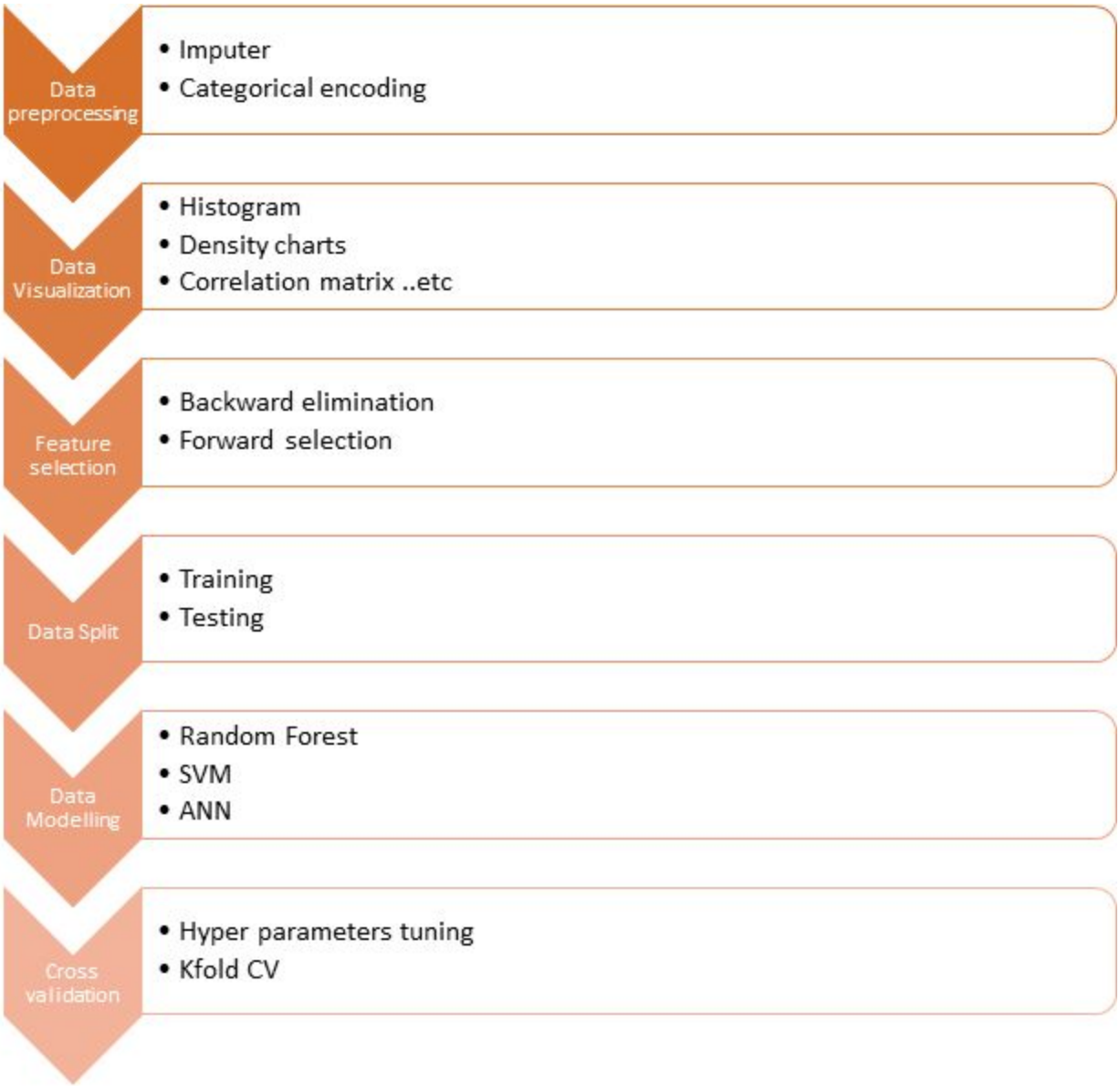
The main objective of this task is to predict the dependent variable y from the independent variables x0-x11.The below shows a snapshot of the dataset given

x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	y
0.878866	0.074382	0	0.678118	6	1	0.616409	1	0.640969	1	0.60315	0	1.403795
0.78621	0.731166	1	0.300039	8	6	0.196483	1	0.02944	6	0.321964	0	1.427232
0.2717	0.139232	0	0.079731	7	5	0.874135	1	0.720054	2	0.64267	0	0.569813
0.686875	0.231666	1	0.572523	3	0	0.873219	1	0.965357	8	1.004235	0	1.208943
0.970329	0.546017	0	0.963757	6	7	1.002242	1	0.881175	1	0.95192	1	2.832496
0.455573	0.45727	0	0.687938	6	2	0.817973	1	0.508673	6	0.228743	0	0.849171
0.705702	0.35374	0	0.210857	2	5	0.138315	1	0.808802	7	0.689062	0	1.549637
0.890946	0.830651	1	0.412818	6	5	0.818191	0	0.373028	4	0.545431	0	1.520959
1.001895	0.512808	0	0.278048	15	5	0.448449	1	0.520652	1	0.279179	0	2.626737
0.586803	0.341806	0	0.269169	3	2	0.705169	1	0.254609	1	0.331992	0	0.488805
0.300702	0.237832	1	0.533015	7	4	0.423871	1	0.151054	6	0.243896	0	0.585224
0.790844	0.16939	1	0.370598	12	1	0.262685	1	0.894947	7	0.375576	0	1.713432
0.339651	0.9301	1	0.50276	2	5	0.849879	1	0.944145	5	0.990294	0	2.082577
0.485803	0.517424	1	0.243048	7	3	1.017658	0	0.987487	0	0.474738	0	1.315977
0.809748	0.499947	1	0.225806	16	3	0.694246	1	0.723335	6	0.824967	1	2.678614
0.403601	0.304654	1	0.499892	14	1	0.63241	0	0.151267	1	0.524917	0	0.927961
0.050797	0.529004	1	0.260553	13	5	0.287789	0	0.243285	4	0.703746	0	0.799217
0.855796	0.624102	1	0.156746	10	8	0.415603	1	0.253515	3	0.354165	0	1.864249
0.250709	0.04844	1	0.088684	5	6	0.963096	1	0.187436	2	0.229636	0	0.446331
0.238762	0.809599	0	0.462725	8	5	0.602688	0	0.520293	2	0.653021	0	1.084922
0.669932	0.937916	1	0.820858	8	3	0.52697	0	0.207177	0	0.995721	0	1.178086
0.352216	1.017994	1	0.687506	16	10	0.235665	1	0.740502	7	0.952443	0	3.13768
0.308477	0.525392	0	0.457285	10	1	0.321997	0	0.366617	7	0.300273	0	0.805051
0.654299	0.489497	0	0.543149	15	0	0.425334	0	0.692847	1	0.728054	0	1.782054
0.360638	0.134896	1	0.1825	3	5	0.598958	1	0.98327	4	0.734067	0	0.752749
0.339617	0.01075	0	0.75317	11	5	0.358009	0	0.688814	7	0.273903	0	0.841739
0.806391	0.201163	0	0.896912	10	0	0.388657	1	0.813855	1	0.941472	0	2.085192

Fig: Dataset description

From the dataset given we resort to applying regression techniques to predict the value of the independent variable. Based on the dataset given we need to determine whether we would need to apply linear or nonlinear regression techniques to solve our problem.

Process Flow model



Preprocessing

Our dataset needs to be preprocessed before we try to fit to various regression models. Effective preprocessing techniques could help to improve the performance our models.

Some common preprocessing elements include the following

Imputing missing values - In many situations a dataset is impacted by the presence of null values. Effective imputing of these null values can help improve the performance of our models. SOme of the common null imputations strategies followed are mean, median and most frequent. Based on these strategies null values are effectively imputed so that the performance of our results are not impacted by the presence of null values.

Categorical Encoding - In many situations the dataset contains features that are of categorical types. In these situations the categories need to be encoded so that everything in the dataset is in numerical format.
Label encoder and Onehotencoder are the encoders that are mostly used together to encode categorical variables. Once that is done care is taken to avoid the dummy variable trap. One of the columns is removed so as to avoid the dummy variable trap.

Feature scaling

The final step of preprocessing includes feature scaling. This involves transforming all features to a common scale. This ensures that there is no greater influence of one feature on the target variable that the other, i.e bias arising based on the value of the numerical value would be avoided with the help of scaling.

In our dataset there are no nulls and hence there is no need for imputing missing variables. There are no categorical variables as confirmed earlier through the given description of the dataset. The features need to be scaled so that they are on a same scale. Each feature is transformed so that it lies within a certain range. It translates the features to a range between 0 and 1. Some packages have an inbuilt scaling criterion that automatically scales the features and reduces complexity. Scaling is necessary before building our machine learning model , because in many situations the euclidean distance between points may seem or lead to one point dominating the other and consequently more influencing the value of the target variable. This anomaly can be avoided so that all the features are transformed to a particular scale and then used to build our machine learning model.

Visualization

We try to visualize the dataset to get better intuitions about the dataset. We need to visualize the correlation between the variables to get a better understanding of our dataset.

Plot 1: Histograms

One of the most simplest and most commonly used univariate plot is the histogram. Histograms help to understand the distribution of each attribute in the dataset.

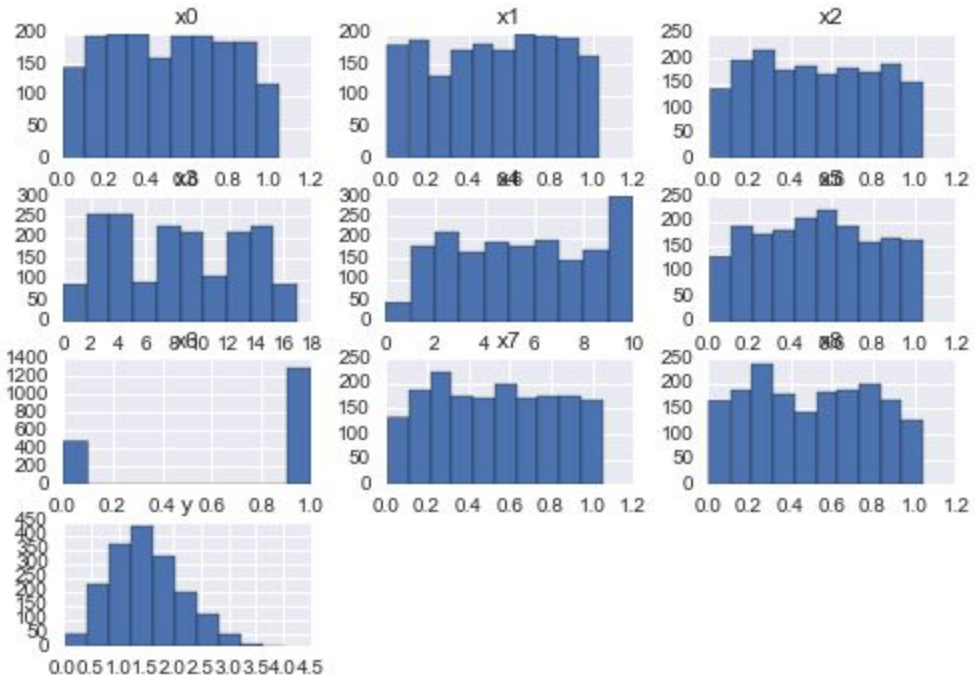


Fig: Histograms

The histograms show information about each feature whether its is normally distributed, skewed to the left or skewed to the right. The independant variable is almost symmetric in nature.X7 is close to left skewness. These figures help to analyze our features better and help us choose better models.

Plot 2 : Box plots

Box plot is also another effective way to summarize the distribution of each feature in the dataset. Box plots are effective since they give a better intuition about the median, the 25th and 75 th percentiles and also information about the amount of outliers. The points that lie on the other side of the whiskers are called outliers

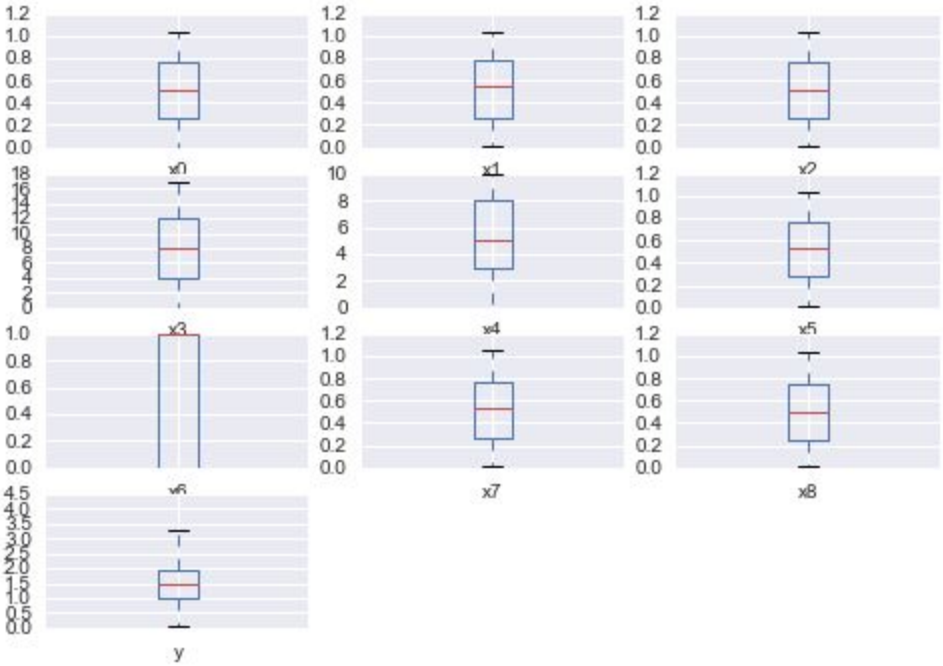
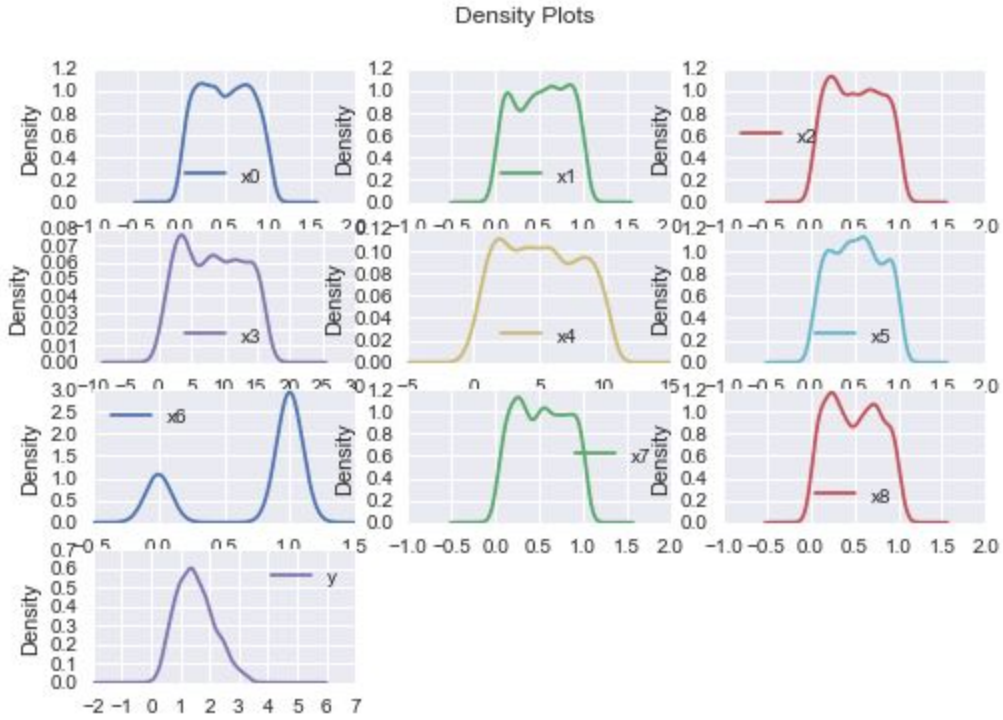


Fig : Box Plots

Plot 3 - Density Plots

Density plots are also univariate plots that give an idea about distribution of each feature of the dataset. With density plot we can see a smooth curve drawn on top of each bin of the histogram that we created previously.



Plot 4: Correlation matrix

Correlation is nothing but a factor that gives us an indication about how related the changes are between two variables of a dataset. A correlation matrix describes the relation between any pair of variables gives us a better understanding about whether variables are positively correlated or negatively correlated. If one variable goes down as the other goes up then they are negatively correlated. If both go up or down in the same direction then they are positively correlated. Correlation can provide a better intuition about how our regression model may perform when fitted. Highly correlated input variables can sometimes affect the performance of certain algorithms.

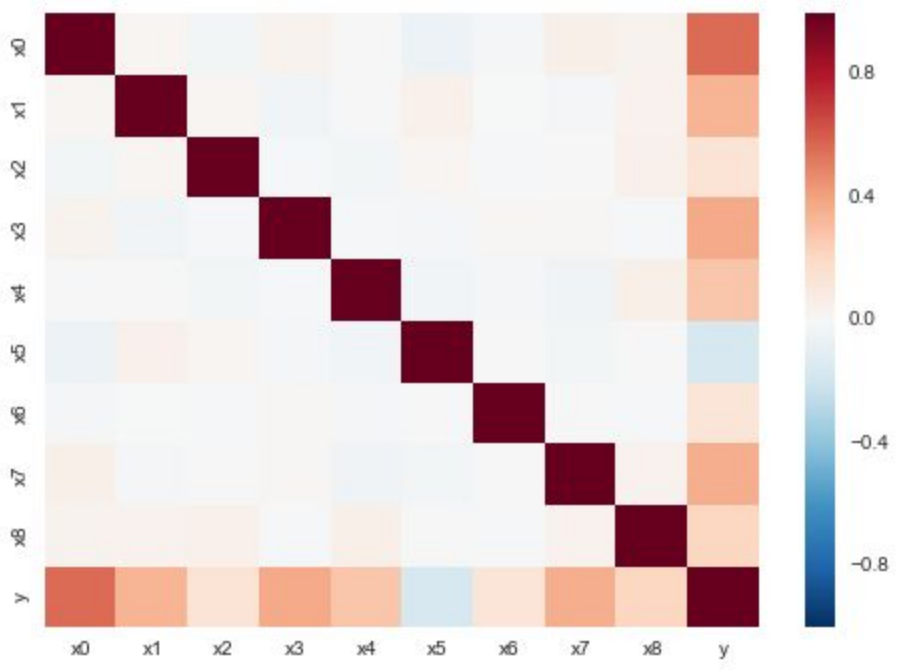


Fig4 : Correlation

From the correlation matrix visualization above it is clear that there is not much correlation between the inputs. Hence we can say that the features are modtly independent of each other.

Feature Selection

Typically there are five renowned methods in feature selection

1. All in
2. Backward elimination
3. Forward selection
4. Bidirectional Elimination
5. Score comparison

Backward elimination

For the purpose of our multivariate dataset we stick to backward elimination since that is the most common method of feature selection that works well in most cases. Backward elimination is mainly based on the significance level which is the p value. P value is a number between 0 and 1 and for the purpose of

feature selection p values are very useful in eliminating features that do not have a significant impact on the target variable. The p value requirement is set to 0.05, this is a usual rule used in most feature selection. However the value of the significance value is not a hard rule and it can be decided based on the dataset. For the purpose of our backward elimination we set our significance level to 0.05. For our purpose of feature selection we use Stochastic Frontier Analysis (SFA) models using Ordinary Least Squares method.

We follow the following steps sequentially

1. We fit the independent features on to our model.
2. Pick the predictor with highest p value ,if pval> 0.05 go to next step else stop process , features have been successfully eliminated
3. Remove predictor with highest pvalue
4. Fit model with the remaining variables, go to step 2

Data split

The data is often split into 3 partitions

1. Train data - The data with which the model trains on
2. Validation data- The data which determines how well the model has been trained
3. Test data- The fresh data which has been kept away and introduced at the last phase to see how well the trained model behaves to new data

For the purpose of our model evaluation we would use train and test data. The main reason for not using the validation set is that we would be using GridsearchCV which is a hyper parameter tuning module. This module has an in built cross validation mechanism, since by default the cross validation for this module is 3 , there would be no need to determine how well the model is trained with the help of a validation set.

1. The data is first divided into train and test sets with approximately 80% and 20% split respectively.
2. Next we fit the necessary models with the train data.
3. Now the regression model is fitted to gridsearchcv with a variety of parameters that could be fitted to this regressor.
4. The gridsearchcv does a thorough optimization of the hyper parameters for this regressor. Once the best parameters are got we fit the regressor with the train dataset and evaluate it with the help of the test
5. Once the mean square error , R squared and Adjusted R squared values are got, the value of the hyperparameters are applied to a more standardised dataset.
6. We evaluate the regressor again with the help of the 10 fold cross validation system
7. Different regressors are evaluated based on different hyper parameters and the best method is selected based on the MSE, Rsquared and Adjysted R squared values.

Machine learning models used

For the purpose of predicting the target variable we should select an optimal machine learning that better fits our dataset.

Support Vector Machines

Support vector machines are popular learning algorithms that are known to work well in classification and regression problems.In addition to performing linear regression and classification SVM have known to work well on non linear data as well. For classification for linearly separable data there can be many different hyperplanes that can accurately separate the data. The problem here would be to find a

hyperplane(margin) that could maximise the separation between two classes. The support vectors in SVM are points that are closer to these margins. The margin of the separator is distance between the support vectors on either side of the separators. SVM uses hard margins or soft margins based on whether the given data is linearly separable or non linear in nature.

Kernel trick

One of the most common terms associated with SVM is the kernel trick in which we have a mapping that can map non linear ,low dimensional data to a linear high dimensional space. The assumption is that non linear data when mapped to a high dimensional space are linearly separable.

SVR

In case of regression SVR is a non parametric regression technique that relies solely on the kernel functions. The goal of the SVR to find a function f(x) that deviates from yn by a value that is no greater that ε for each of the training point in our dataset and at the same time remains as flat as possible.
Since our dataset is a multivariate supervised dataset some of the kernels that we would want to compare for regression could be linear, polynomial and radial basis function. Polynomial regression is a linear regression model that can fit nonlinear data, while radial basis function is used mostly in working with non linear data and is proved to give robust results in most SVM regressions.
We use gridsearchviewCV to evaluate the possible kernels linear , poly and rbf. The idea here is to evaluate the performance of these kernels on our data and also evaluate different epsilon parameters which are the penalty parameters for the error term. For kernels such as poly and rbf there is also a kernel coefficient parameter gamma. Hence there is a need to search over these parameters to break ties between the kernels and find the kernels that can optimally predict the results of our data.

By applying linear, poly and rbf we find that the rbf kernel has been the best SVR kernel that better predicts the results of our data. We evaluate their performance based on the R-squared scoring mechanism. From the gridsearchcv module that we have in python we evaluate the performance of our model over different kernels, epsilon (C) values and gamma parameters.
The optimal results were for the following parameters

'C': 2000
'gamma': 0.01
'kernel': 'rbf'

We then apply these parameters to our SVR and use our training to fit the SVR model. We obtain considerably very good results after this hyper parameter tuning. The metrics that we used to evaluate the performance of our models are R-Squared, Mean Square error and Adjusted R Squared. SInce from the results we see that the data looks to be non linear ,we would weigh the value of adjusted R squared more, since Adjusted R squared metrics better evauluate the performance of a model when dealing with nonlinear data.
The values got for this model are as below

MSE	R squared	Adjusted R squared	10 fold Cross validation	Time taken (seconds)
0.0084	0.981	0.961	0.983	128

From the above results it can be seen that the kernel function rbf can be seen to better fit our data. The value of adjusted R squared is extremely high and hence we could consider this as our baseline model. It should be noted that the value of epsilon should not be too high as the model may overfit the data for higher penalties imposed on errors. Hence we evaluate the performance of our model using a more standardised dataset using 10 fold cross validation. The cross validation results are considerably high as well making it clear that our results are more generalized and are capable of accurately predicting new inputs that would be fed to this model in the future.

Visualization

The below figures visualize the decision boundaries of SVR for train and test datasets respectively

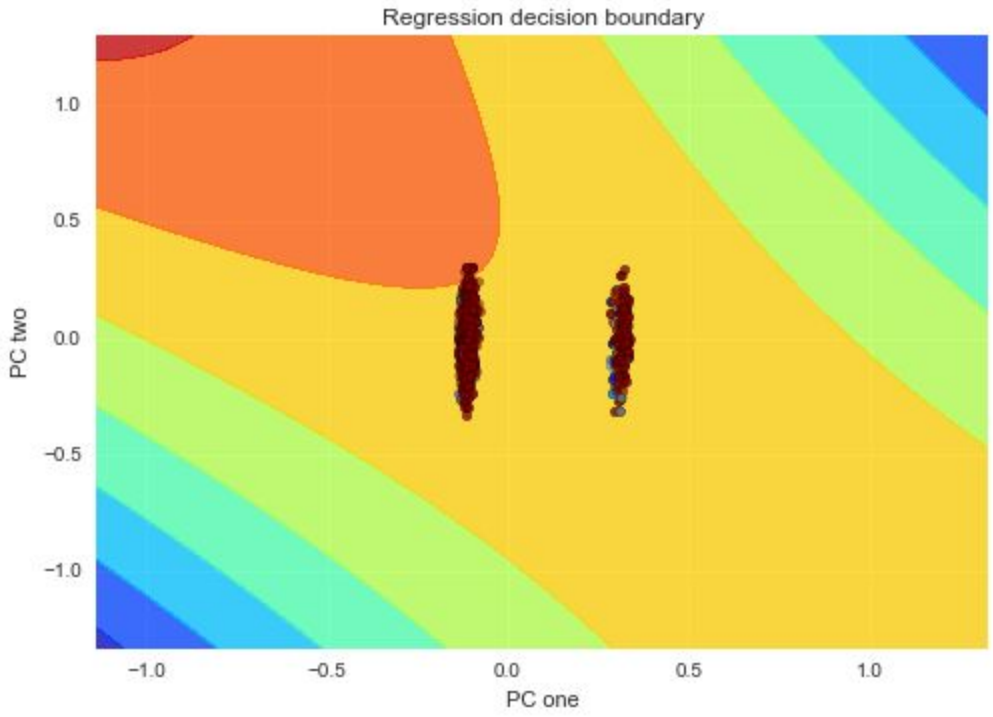


Fig : Training set

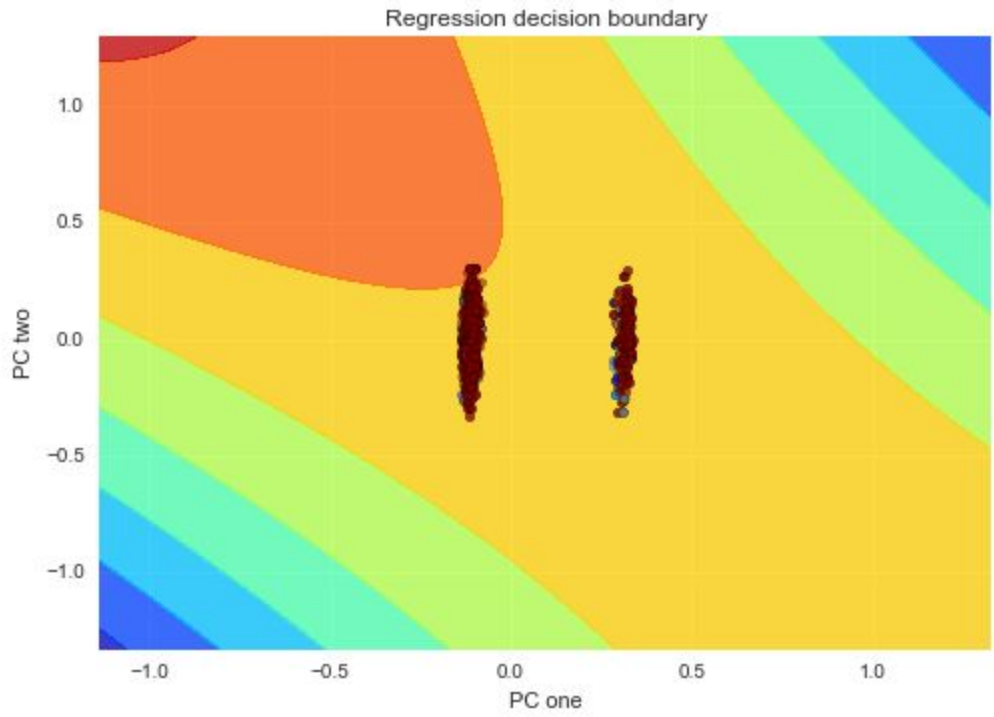


Fig : Test set

Random Forest

Random forest is also a powerful ensemble learning technique that is used in classification and regression tasks. Random forest models are constructed by constructing a collection of number of decision trees. Decision trees are often referred to as off the shelf classifiers as they are easy to use without much preprocessing. The decision trees divide the decision boundaries as axis parallel rectangles.The idea here is to construct a collection of trees with controlled variance. The problem with decision trees as such is that they tend to overfit the training data. The most common technique used to avoid this is reduced error pruning. Random forest follows the approach in which multiple deep decision trees are averaged out so that the output of the random forest remains as generalized as possible. The random forest model tend to reduce the variance in decision tree techniques and hence avoids overfitting on training data

The random forest model is parameter tuned with the help of gridsearchcv. The n_estimators and max_features are the parameters that are being tuned for this purpose. In most situations as the n_estimators increases the random forest model tends to be more optimized. The max_features parameter looks at the maximum features to be considered at each split. The possible parameters for max_features are auto, sqrt and log2. The optimal parameter for our dataset is log2 which is the log of number of features.

The results for this model are summarized as below

MSE	R squared	Adjusted R squared	10 fold Cross validation	Time taken (seconds)
0.028	0.935	0.872	0.947	123

From the above results it can be seen that our model gives a very good result with randomforestregression. Eventhough the R squared value and the Adjusted R squared values are not as high as our baseline SVR model, this model also gives out considerably good results. The random forest model is a touch faster that our baseline model.

Visualization

The below figures show a visualization of the train and test sets of our random forest model. As seen below the decision trees divide the decision boundaries into axis parallel rectangles.

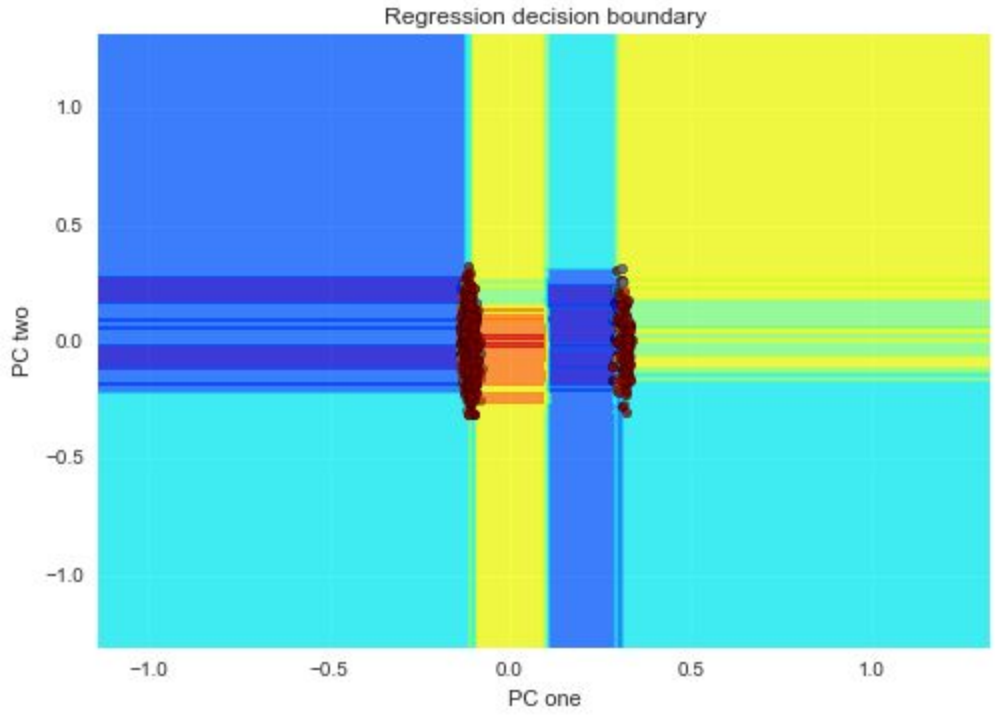


Fig : Training set

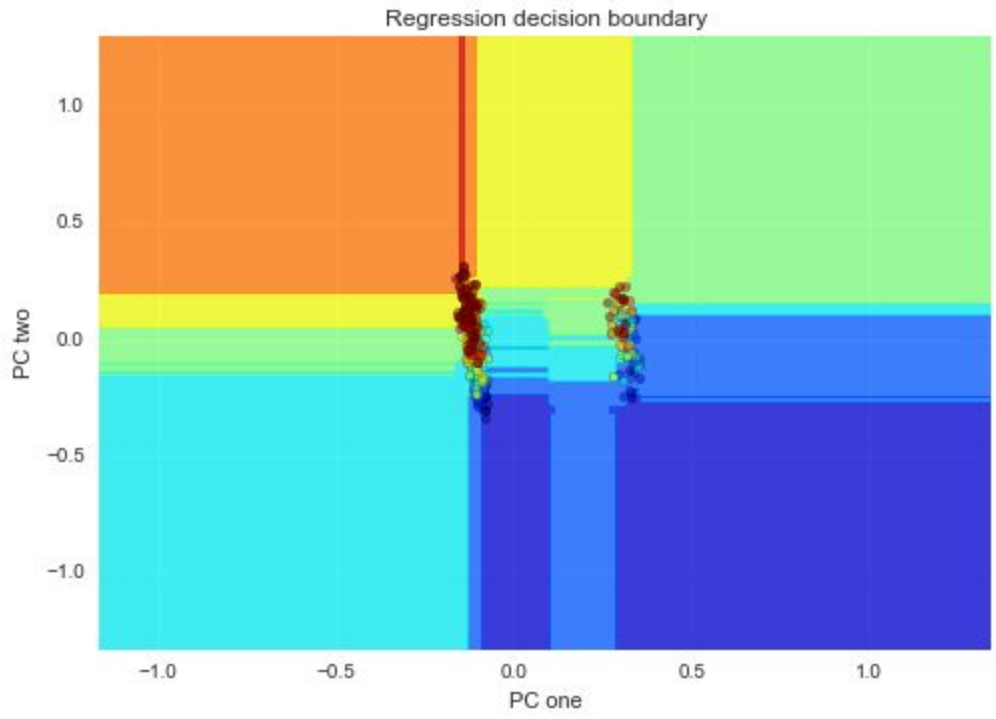


Fig: Test set

Artificial Neural Networks

One of the most effective computational models used in supervised learning regression and classification tasks is the artificial neural networks. ANNs mainly work with the help of three layers input layer, hidden layer and the output layers. The performance of the ANN depends on the performance of the hidden layers. The main objective of the ANN is to mimic the functionality of the human brain. The task of creating an effective neural network is one of the rapidly growing fields in Deep learning.

For our purpose we use Keras which is an open source neural network library available in python. The keras that was used for our method runs on top of Tensorflow backend. The neural network is trained using Stochastic gradient descent optimizer. The following are the sequence of steps that were followed in constructing our neural network

1. Random initialization of weights to numbers close to zero
2. Input first observation of the dataset into the input layer. Each feature of the dataset should correspond to each input node.
3. Forward propagation from left to right such that the neurons are activated so that the weights limit the impact of each neuron's activation
4. Continue forward propagation until getting the predicted result y
5. Compare output y with the actual y and estimate the error
6. Backpropagation from right to left so that the weights get updated based on their influence on the error.
7. The updating of weights is decided by the learning rate
8. Repeat 1 to 7 after a batch of operations

9. When the whole training set has been fed into the neural network that makes an epoch, use the epoch parameter to decide on the number of epochs for you operation.

The keras sequential regressor is used for the purpose of our evaluation. The aprameters for the ANN regressor are described below.

out_dim - number of nodes in input layer
init - Initialize weights to small numbers close to 0
(Glorot with weights sampled from the Uniform distribution)
activation - rectifier function , relu
input_dim - number of input or independent features

The optimizer used for the purpose of stochastic gradient descent is called Adam. Adam is one of the more common and efficient optimizers used in neural networks. The results that were got from our neural networks model are summarized below.

MSE	R squared	Adjusted R squared	10 fold Cross validation	Time taken (seconds)
0.009	0.98	0.9555	0.9697	434

Results summary

Model	**MSE**	**R squared**	**Adjusted R squared**	**10 fold Cross validation**	**Time taken (seconds)**
SVR	0.0084	0.981	0.961	0.983	128
Random Forest	0.028	0.935	0.872	0.947	123
ANN	0.009	0.98	0.9555	0.9697	434

From the above models the following intuitions can be inferred

1. Our fastest model is the random forest model
2. The most accurate model is our SVR
3. The ANN is the second best model

The above prediction method collectively performs well. Random forest may improve performance if tuned with more hyper parameter values. The overall accuracy of all three models were very good the neural networks method collectively outperformed other two models

SVM better performance

The reason why SVM performs better than ANN is due to the fact that SVM is global and unique while ANN can be affected by multiple local minima. ANNs depend on the dimensional complexity of the input space while SVMs do not depend on those factors as they are simple and quick. SVM are less prone to overfitting and at most times perform better. These could be the possible reasons as to why the SVM model outperformed ANN in our model