RUTGERS UNIVERSITY

COMPUTER SCIENCE

CS 536 : Machine Learning

Final Project - Data Completion and Interpolation (ML3 Dataset)

Author:

Sagar Jain- sj735

Aayush Mandhayan- am2447

Supervisor:

Professor Wes Cowan

11 May 2019

Question 1: Describe your model:

We considered the ML3 data set for our Final ML project.

One of the most common problems we have faced in Data Cleaning/Exploratory Analysis is handling the missing values. Firstly, understand that there is NO good way to deal with missing data. we have come across different solutions for data imputation depending on the kind of problem — Time series Analysis, ML, Regression etc. and it is difficult to provide a general solution.

Imputation vs Removing Data

Before jumping to the methods of data imputation, we tried to understand the reason why data goes missing.

1. **Missing at Random (MAR):** Missing at random means that the propensity for a data point to be missing is not related to the missing data, but it is related to some of the observed data.

2. **Missing Completely at Random (MCAR):** The fact that a certain value is missing has nothing to do with its hypothetical value and with the values of other variables.

3. **Missing not at Random (MNAR):** Two possible reasons are that the missing value depends on the hypothetical value (e.g. People with high salaries generally do not want to reveal their incomes in surveys) or missing value is dependent on some other variable's value (e.g. Let's assume that females generally don't want to reveal their ages! Here the missing value in age variable is impacted by gender variable)

In the first two cases, it is safe to remove the data with missing values depending upon their occurrences, while in the third case removing observations with missing values can produce a bias in the model. So, we have to be really careful before removing observations. Note that imputation does not necessarily give better results.

Deletion:

Since, we need to train the data over our model and we need to make sure that we chose appropriate features as the input, at first, we removed all the features where more than 50 percent of the data was missing only if that variable is insignificant. Having said that, imputation is always a preferred choice over dropping variables.

List of Deleted Columns:

{'SomeEndorse','MostEndorse','worstgrade_order','tempcomm_order','tempcomf_order','tempagenm_order','tempagenf_order','mcsome_order','mcmost_order','lowpower_order','highpower_order','elmweak_order','elmstrong_order','consent_order','compcode_order','bestgrade_order','Pool20','Pool16c','Pool7e','Pool7a','Pool5b','Pool2e','Pool1','Notes','SRTFCorrect','worstgrade5','worstgrade4','worstgrade3,'worstgrade2','worstgrade1','anagrams1', 'anagrams2', 'anagrams3', 'anagrams4', 'attention', 'attentioncorrect', 'bestgrade1', 'bestgrade2','bestgrade3', 'bestgrade4', 'bestgrade5', 'lowpower', 'mcmost1', 'mcmost2', 'mcmost3', 'mcmost4', 'mcmost5','mcsome1', 'mcsome2', 'mcsome3', 'mcsome4', 'mcsome5'}

which reduced our features to 210 features total.

Secondly, since almost 98 percent of the data was numeric data and we were facing many problems if data was in String format as we were not able to convert it into float, so we need to remove all the features from our dataset where String data was used.

We actually considered these features while performing NLP model on them later in our pipeline.

#dropping columns with text data

drop_col = ['ethnicity', 'Site', 'major', 'tempest1', 'Date.x', 'Experimenter', 'OrderofTasks', 'SRCondition', 'SRMeetingResponse', 'StartDate.x', 'EndDate', 'ClipBoardMaterial', 'DateComputer', 'TimeComputer', 'AttentionCheck', 'PowerCond', 'CredCond', 'Genderfactor', 'SubDistCond', 'TempCond4', 'TempCond', 'TargetGender']

#df = df.drop(columns=drop_col)

len(drop_col)

That reduced our features further to 182features total.

Thirdly, we had to remove the rows of data's where more than 70 percent of the features were missing, as our prediction models were giving vague and inappropriate outcomes for these features.

So the final dimensionality of our data set was reduced to 2350 rows and 186 features.

Imputation:

Computing the overall mean, median or mode is a very basic imputation method, it is the only tested function that takes no advantage of the time series characteristics or relationship between the variables. It is very fast, but has clear disadvantages. One disadvantage is that mean imputation reduces variance in the dataset.

We classified our features into categorical and numerical feature type before training them in our model.

For the missing values imputation, At First, we trained our data by inserting the mean value in the case for numerical features and mode value in the case of categorical features.

We also tried by imputing the zero value for the missing data values but found that our model behaved in more or less the same way for both the imputation.

Conclusion:

If the features were categorical, then we used the mode values as imputed data and mean values for numerical feature data.

Our Approach:

For training the model:

- We removed all the rows from the dataframe wherever our target value was holding a imputed value.
- Split the dataframe into training and test dataset in the ratio 4:1 to train the model.
- We performed cross validation for 5 folds using above split.

For Predicting:

- We took the dataframe which holds all the data rows which we removed from the training dataset with imputed values for target feature inorder to get prediction based on accuracy of the model.

Models Used:

Ridge, Lasso Regression & Random Forest - For Numerical/Continuous Features Type.

Random Forest and SVM- For Categorical Data Type.

#Our Understanding while choosing these models:

Regression Model Analysis:

Regularization:

This is a form of regression, that constrains/regularizes or shrinks the coefficient estimates towards zero. In other words, **this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.**

A simple relation for linear regression looks like this. Here Y represents the learned relation and *β represents the coefficient estimates for different variables or predictors(X).*

Y ≈ β0 + β1X1 + β2X2 + …+ βpXp

The fitting procedure involves a loss function, known as residual sum of squares or RSS. The coefficients are chosen, such that they minimize this loss function.

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Now, this will adjust the coefficients based on our training data. *If there is noise in the training data, then the estimated coefficients won't generalize well to the future data. This is where regularization comes in and shrinks or regularizes these learned estimates towards zero.*

Ridge Regression

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

Above image shows ridge regression, where the **RSS is modified by adding the shrinkage quantity.** Now, the coefficients are estimated by minimizing this function. Here, **λ is the tuning parameter that decides how much we want to penalize the flexibility of our model.** The increase in flexibility of a model is represented by increase in its coefficients, and if we want to minimize the above function, then these coefficients need to be small. This is how the Ridge regression technique prevents coefficients from rising too high.

Learning Points:

- It shrinks the parameters, therefore it is mostly used to prevent multicollinearity.
- It reduces the model complexity by coefficient shrinkage.
- It uses L2 regularization technique.

Lasso

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| \;=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

Lasso is another variation, in which the above function is minimized. It's clear that **this variation differs from ridge regression only in penalizing the high coefficients**. It uses |βj|(modulus)instead of squares of β, as its penalty.

Learning Points:

- It uses L1 regularization technique (will be discussed later in this article)
- It is generally used when we have more number of features, because it automatically does feature selection.

What does Regularization achieve?

A standard least squares model tends to have some variance in it, i.e. this model won't generalize well for a data set different than its training data. ***Regularization, significantly reduces the variance of the model, without substantial increase in its bias***. So the tuning parameter λ, used in the regularization techniques described above, controls the impact on bias and variance. As the value of λ rises, it reduces the value of coefficients and thus reducing the variance. ***Till a point, this increase in λ is beneficial as it is only reducing the variance (hence avoiding overfitting), without losing any important properties in the data.*** But after certain value, the model starts losing important properties, giving rise to bias in the model and thus underfitting. Therefore, we chosed the value of λ carefully.

We trained our model using both Lasso and Ridge Regression techniques and we will share the results in subsequent sections.

Classification Model Analysis

SVM:

The support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

Tuning parameters: Kernel

Kernel

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

f(x) = B(0) + sum(ai * (x,xi))

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

The **polynomial kernel** can be written as

$K(x,xi) = 1 + sum(x * xi)^d$ and

exponential as $K(x,xi) = exp(-gamma * sum((x - xi^2))$.

*Polynomial and exponential kernels calculate the separation line in higher dimension. This is called **kernel trick***

Random Forest:

Random Forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The "forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Basically, Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current

machine learning systems. We took random forest in classification, since classification is sometimes considered the building block of machine learning.

Since we were dealing with Categorical feature Data, we chosed Random Forest over any other model.

Question 2: Describe your Training Algorithm: Given your model, how did you fit it to the data? What design choices did you make, and why? Were you able to train over all features? What kinds of computational tradeoffs did you face, and how did you settle them?

List of Training Algorithms used:

- For whole data set: Variational Auto Encoders
- Alternatively:
    - For Numerical Data: Linear Regression/Lasso Regression/Ridge Regression
    - For Categorical Data: Random Forest/SVM

Variational Auto Encoders:

An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal “noise”. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. Recently, the autoencoder concept has become more widely used for learning generative models of data.

An autoencoder learns to compress data from the input layer into a short code, and then uncompress that code into something that closely matches the original data. This forces the autoencoder to engage in dimensionality reduction, for example by learning how to ignore noise. Some architectures use stacked sparse autoencoder layers for image recognition.[citation needed] The first encoding layer might learn to encode basic features such as corners, the second to analyze the first layer's output and then encode less local features like the tip of a nose, the third might encode a whole nose, etc., until the final encoding layer encodes the whole image into a code that matches (for example) the concept of "cat". The decoding layers learn to decode the representation back into its original form as closely as possible. An alternative use is as a generative model: for example, if a system is manually fed the codes it has learned for "cat" and "flying", it may attempt to generate an image of a flying cat, even if it has never seen a flying cat before.

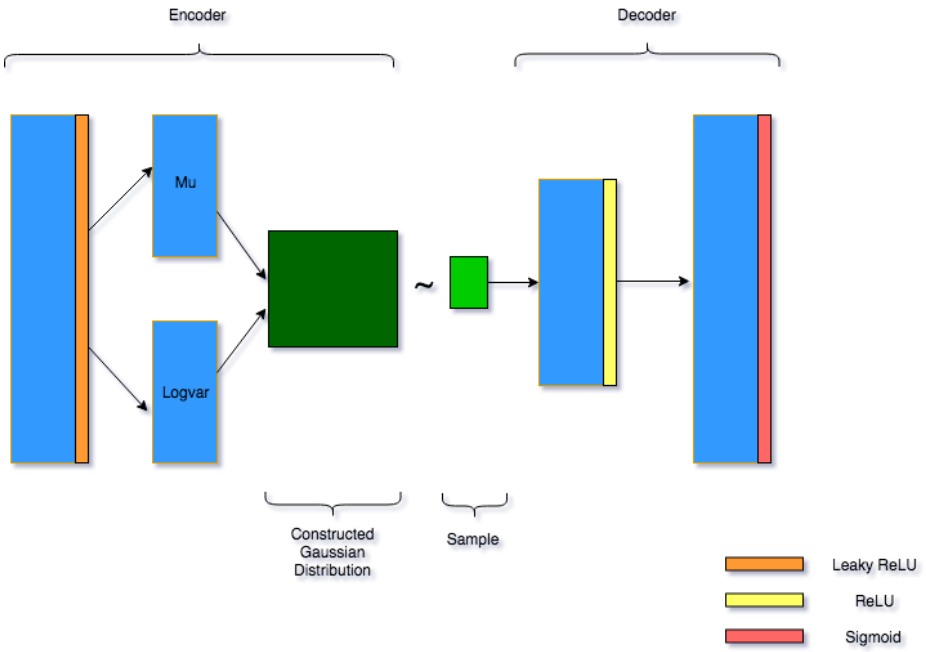


Figure: Sample Architecture of a Variational Autoencoders.

Fit data into VAE:

Steps:

1. Convert the categorical variables using one-hot encoding and save them in the preprocessed data frame.
2. We take the above data frame and push that to the VAE architecture.
3. We tried our VAE with various architectures to improve performance, structures used:
    a. 5 layered (Best RMSE)
        i. inputs of each layer as [6015, 3000, 1000, 3000, 6015]
        ii. Number of features (values in encoded data row) 6015.
        iii. Batch size: 64 (should be of power 2, as it improves computational perfomance)
        iv. Step size:1000
        v. RMSE: 576009
        vi. Computational Time: 3hr.
    b. 5 layered
        i. inputs of each layer as [6015, 3000, 1000, 3000, 6015]
        ii. Number of features (values in encoded data row) 6015.
        iii. Batch size: 64
        iv. Step size:100
        v. RMSE: 659831
        vi. Computational Time: 1hr 30 min.

   c. 3 layered
      i. inputs of each layer as [6015, 3000, 6015]
      ii. Number of features (values in encoded data row) 6015.
      iii. Batch size: 64
      iv. Step size:100
      v. RMSE: 769861
      vi. Computational Time: 1hr.
4. Decode one hot encoded variable from the output data.

We use all the features from the data set (after removing columns original data where 50 % of column data is missing).

We tried to train VAE for more than 1000 steps, but due to time constraints and the poor RMSE output we did not proceed further with this model.

Alternative Approach: Mixture of Models

In this approach, we focus on categorical data first and then on numerical data. For both, numerical and categorical data we try out different models to find which one has the best accuracy or least RMSE value and use that model to impute the missing data. We look at approaches taken for both the data below:

Categorical Columns:

Fit models:

1. Convert the categorical variables using one-hot encoding and save them in the preprocessed data fame.
2. Normalize the data set.
3. We take the above data frame and push that to below models:
4. We take two approaches on both the classification model we trained:
   a. We use the predicted value for a column to be used for prediction of next column.
   b. We don't use the predicted value for the previous columns.
5. Classification models:
   a. Random Forest:
      i. We created a Cross-Validation model which takes the train/test split size, count of cross validations and number of trees for random forest model. And we judge the accuracy of our model using True positive and True Negative count.
   b. Support Vector Machine:
      i. We created a Cross-Validation model which takes the train/test split size, count of cross validations and step count for SVM model. And we judge the accuracy of our model using True positive and True Negative count.
6. We pick the best model above, in our case we got better performance using Random Forest for 500 trees

7. We use this to predict the missing data for the target feature.
8. Scale back the normalized data.
9. Decode the one-hot-encoded data.

Numerical Columns:

Fit models:

1. Convert the categorical variables using one-hot encoding and save them in the preprocessed data fame.
2. Normalize the data set.
3. We take the above data frame and push that to below models:
4. We take two approaches on both the regression model we trained:
    a. We use the predicted value for a column to be used for prediction of next column.
    b. We don't use the predicted value for the previous columns.
5. Classification models:
    a. Ridge Regression:
        i. We created a Cross-Validation model which takes the train/test split size, count of cross validations and lambda value for ridge regression. And we judge the accuracy of our model using the least RMSE.
    b. Lasso Regression:
        i. We created a Cross-Validation model which takes the train/test split size, count of cross validations and lambda value for lasso regression. And we judge the accuracy of our model using the least RMSE.
    c. Random Forest:
        i. We created a Cross-Validation model which takes the train/test split size, count of cross validations and number of trees for random forest model. And we judge the accuracy of our model using RMSE.
6. We pick the best model above, in our case we got better performance using Ridge Regression for lambda = 0.02
7. We use this to predict the missing data for the target feature.
8. Scale back the normalized features.
9. Decode the one-hot-encoded data.

We were able to train our model using all the preprocessed data frame. But it took a considerable amount of time. To reduce the computational time, we used PCA algorithm to reduce dimensionality of our data. Which in turn reduced the computational time and improved model accuracy by a small amount.

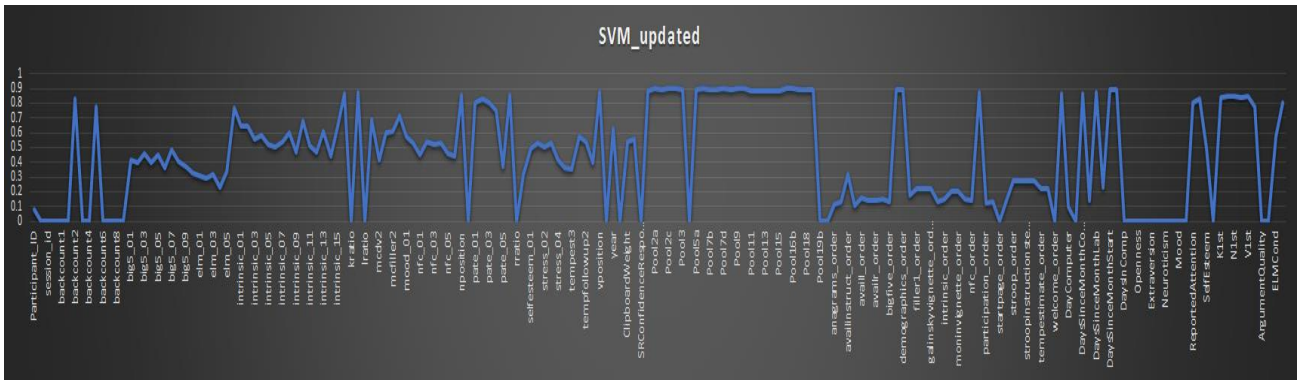
Question 3: Describe your Model Validation: How did you try to avoid overfitting the data? How did you handle the modest (in ML terms) size of the data set?

In statistics, overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". An overfitted model is a statistical model that contains more parameters than can be justified by the data. In our model we use the following way to reduce over fitting:

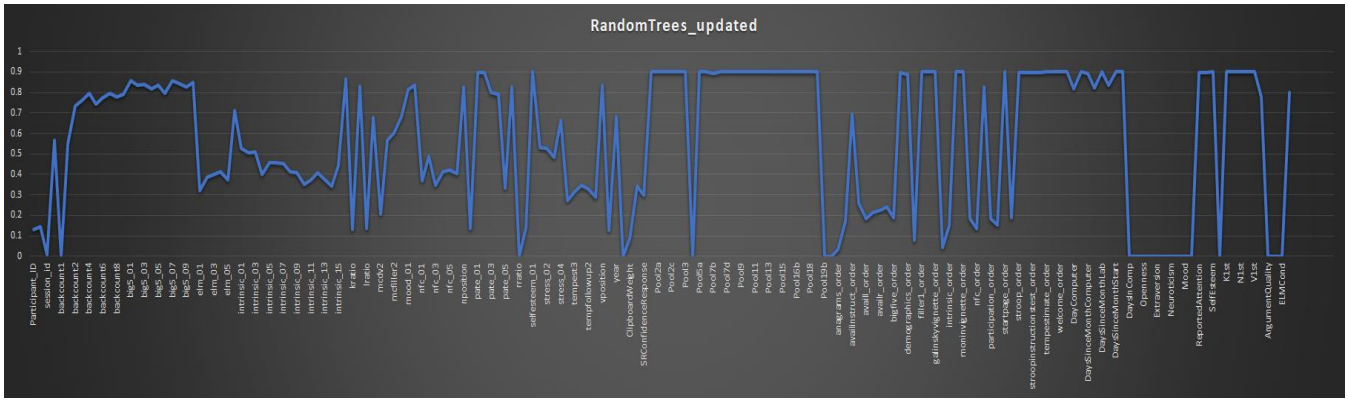
1. K-fold Cross Validation:
    a. Cross-validation is a powerful preventative measure against overfitting. In standard k-fold cross-validation, we partition the data into k subsets, called folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining fold as the test set (called the “holdout fold”). Cross-validation allows you to tune hyperparameters with only your original training set. This allows you to keep your test set as a truly unseen dataset for selecting your final model.
    b. As our data is modest (in ML terms), we use Cross Validation to get as much data as we can to train our model and at the same time test our model without compromising with the accuracy.
2. Regularization:
    a. Regularization term keeps the weights small making the model simpler and avoiding overfitting. λ is the penalty term or regularization parameter which determines how much to penalizes the weights
    b. For numerical data we have used Ridge and Lasso Regression models which internally have L2 and L1 regularization respectively.
3. Normalization:
    a. Normalization is the process of scaling individual samples to have unit norm. This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.
    b. We use Normalization for both Categorical & Numerical Models which improves accuracy & RMSE of the models.
4. Using Ensemble models:
    a. Ensemble models in machine learning combine the decisions from multiple models to improve the overall performance.
    b. We have used Random Forest which is an ensemble model for classification. Which performs better than SVM model we have tried.
5. Feature reduction:
    a. Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.
    b. We use PCA to find features which best describes our data, in order to use these subsets of features instead of each and every one of them to decrease computation cost and help in reducing overfitting on the data.
    c. We also use a feature correlation chart to select features which have considerable amount of correlation for modelling and also keeping in mind the issued caused by the features if they have very high ~.99 correlation.

Our Model Predictions for

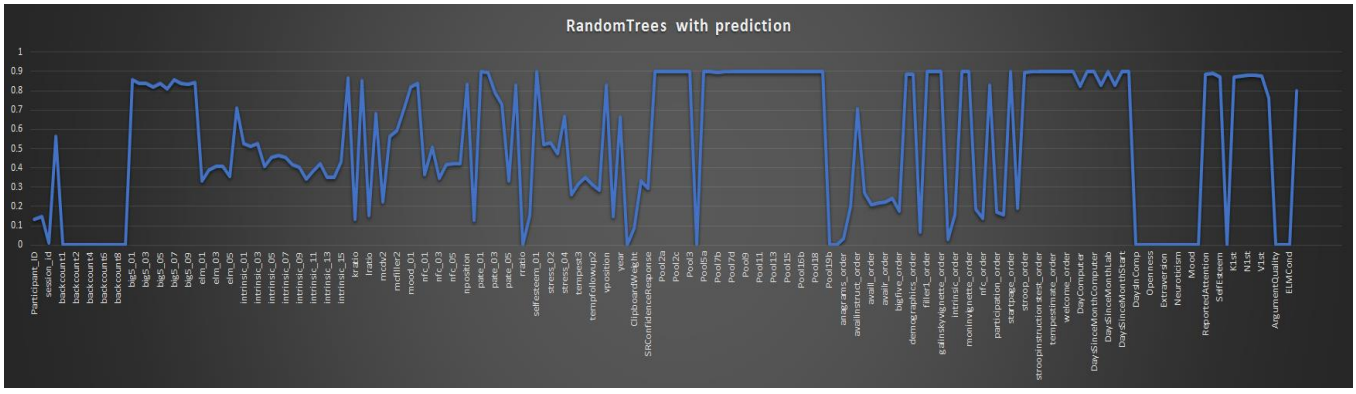
SVM model: It predicted quiet well for categorical features (mainly 0' and 1's value) but failed for continuous numerical features.



Random Forest Regression model: It predicted well for categorical features as well for the continuous numerical features.



Random Forest Regression with Prediction model: We made some changes in our model, while performing prediction we made changes in our test dataset by updating it with the previous predicted feature value for the subsequent feature prediction in-order to build the feature dependency in our model. This resulted in the better prediction result for features data values.



Sharing the Prediction Result set for all the models:

Features	RandomTrees_updated	SVM_updated	RandomTrees with prediction	Ridge Regression	Lasso Regression
Participant_ID	0.131524008	0.083507307	0.131524008	3.24E+174	3.70E+174
RowNumber	0.146137787	0	0.146137787	1.56E+163	1.34E+163
session_id	0.008350731	0	0.008350731	7.95E+166	1.23E+168
age	0.565762004	0	0.565762004	8.10E+160	8.76E+159
backcount1	0	0	0	1.11E+166	1.11E+166
backcount10	0.549060543	0	0.549060543	6.68E+163	7.10E+163
backcount2	0.736534447	0.828601253	0.736534447	1.55E+164	1.91E+164
backcount3	0.75908142	0	0.75908142	5.42E+165	5.41E+165
backcount4	0.797494781	0	0.797494781	1.70E+164	1.69E+164
backcount5	0.744050104	0.774112735	0.744050104	3.12E+162	3.39E+162
backcount6	0.775991649	0	0.775991649	4.16E+166	4.06E+166
backcount7	0.797494781	0	0.797494781	4.92E+165	3.27E+165
backcount8	0.777870564	0	0.777870564	3.19E+166	3.17E+166
backcount9	0.792901879	0	0.792901879	5.56E+162	4.62E+162
big5_01	0.856784969	0.419624217	0.856784969	3.64E+156	1.10E+158
big5_02	0.834237996	0.39874739	0.839874739	1.41E+150	6.68E+158
big5_03	0.839874739	0.459290188	0.837995825	2.72E+150	4.57E+159
big5_04	0.817327766	0.39874739	0.817327766	4.38E+148	9.51E+158
big5_05	0.83611691	0.448851775	0.837995825	2.19E+150	3.64E+159
big5_06	0.794780793	0.35908142	0.807933194	5.78E+156	2.67063E+12
big5_07	0.858663883	0.482254697	0.856784969	2.24E+151	8.96911E+13
big5_08	0.845511482	0.409185804	0.839874739	4.19E+150	2.78E+158
big5_09	0.828601253	0.375782881	0.832359081	9.78E+149	553708509.2
big5_10	0.847390397	0.329853862	0.843632568	7.12E+148	2262118842
elm_01	0.319415449	0.313152401	0.329853862	1.31E+160	1.67E+158
elm_02	0.388308977	0.296450939	0.388308977	1.73E+159	2.51E+157
elm_03	0.39874739	0.317327766	0.407098121	8.25E+159	1.17E+158
elm_04	0.411273486	0.231732777	0.409185804	2.73E+159	5.82E+159
elm_05	0.373695198	0.340292276	0.356993737	2.30E+157	6.87E+159
gender	0.713987474	0.768267223	0.711899791	8.82E+158	101697389.1
intrinsic_01	0.528183716	0.647181628	0.528183716	1.62E+158	1.41E+155
intrinsic_02	0.505219207	0.645093946	0.509394572	1.31E+159	132917683.5
intrinsic_03	0.509394572	0.553235908	0.524008351	6.94E+157	1240204400

intrinsic_04	0.39874739	0.582463466	0.409185804	2.09E+157	1020623067
intrinsic_05	0.457202505	0.519832985	0.457202505	4.78E+158	2001712615
intrinsic_06	0.457202505	0.505219207	0.463465553	3.10E+159	4.53E+158
intrinsic_07	0.45302714	0.536534447	0.455114823	6.00E+158	268084639.6
intrinsic_08	0.413361169	0.603340292	0.417536534	5.66E+158	2.1319E+12
intrinsic_09	0.409185804	0.471816284	0.400835073	1.74E+158	1622349611
intrinsic_10	0.350730689	0.680584551	0.342379958	5.17E+157	6234219787
intrinsic_11	0.371607516	0.513569937	0.386221294	6.88E+159	11194182.46
intrinsic_12	0.409185804	0.465553236	0.419624217	1.13E+159	1.55E+157
intrinsic_13	0.371607516	0.60960334	0.350730689	3.71E+158	1426371928
intrinsic_14	0.340292276	0.442588727	0.352818372	5.74E+158	7.00E+158
intrinsic_15	0.444676409	0.643006263	0.43006263	2.27E+158	6501990161
kposition	0.866179541	0.868058455	0.866179541	3.56E+156	151541111
kratio	0.129436326	0.129436326	0.131524008	1.30E+163	1.25E+163
lposition	0.832359081	0.871816284	0.85302714	1.27E+159	182872886.5
lratio	0.135699374	0.135699374	0.152400835	5.61E+161	7.27E+161
mcdv1	0.678496868	0.693110647	0.682672234	1.95E+159	6.65E+157
mcdv2	0.204592902	0.411273486	0.223382046	9.65E+159	5.97E+159
mcfiller1	0.565762004	0.599164927	0.563674322	1.15E+157	318879023.9
mcfiller2	0.60125261	0.605427975	0.590814196	3.87E+159	1.50E+159
mcfiller3	0.682672234	0.713987474	0.699373695	1.47E+159	2.50E+157
mood_01	0.811691023	0.576200418	0.819206681	2.71E+156	1052559955
mood_02	0.834237996	0.534446764	0.839874739	8.94E+158	593285193
nfc_01	0.369519833	0.448851775	0.363256785	7.45E+156	4.24167E+13
nfc_02	0.488517745	0.536534447	0.507306889	1.82E+158	8.18E+157
nfc_03	0.346555324	0.521920668	0.344467641	9.84E+156	9.27E+158
nfc_04	0.411273486	0.526096033	0.415448852	1.76E+157	1123454140
nfc_05	0.419624217	0.459290188	0.4217119	1.91E+158	1.52E+159
nfc_06	0.405010438	0.444676409	0.423799582	2.66E+158	1.66E+159
nposition	0.828601253	0.860542797	0.832359081	4.16E+157	272908815
nratio	0.133611691	0.133611691	0.129436326	5.33E+161	5.26E+161
pate_01	0.898121086	0.804175365	0.898121086	1.36E+149	2.16E+159

pate_02	0.896242171	0.832359081	0.896242171	1.52E+150	4.13E+159
pate_03	0.798538622	0.802296451	0.791022965	4.48E+157	915742.5806
pate_04	0.791231733	0.747807933	0.729018789	6.54E+157	66480454.54
pate_05	0.334029228	0.375782881	0.331941545	1.73E+159	246108489.6
rposition	0.826722338	0.858663883	0.826722338	1.09E+158	230734341.4
rratio	0.9	0.9	0.9	5.75E+160	1.53E+161
sarcasm	0.139874739	0.317327766	0.152400835	1.96E+160	1.59E+160
selfesteem_01	0.9	0.498956159	0.9	1.11E+148	21664074.41
stress_01	0.532359081	0.528183716	0.519832985	5.25E+158	2.06E+159
stress_02	0.526096033	0.507306889	0.530271399	1.91E+152	2.89E+157
stress_03	0.482254697	0.528183716	0.475991649	2.45E+157	8.00E+158
stress_04	0.661795407	0.411273486	0.668058455	3.31E+158	1.41E+160
tempest2	0.271398747	0.363256785	0.263048017	2.17E+158	3.35E+159
tempest3	0.317327766	0.354906054	0.319415449	4.44E+159	1.43E+159
tempfollowup1	0.348643006	0.576200418	0.352818372	8.49E+157	9.68E+159
tempfollowup2	0.329853862	0.540709812	0.315240084	1.27E+159	2.36E+160
tempfollowup3	0.290187891	0.394572025	0.286012526	2.90E+159	3.82E+159
vposition	0.83611691	0.86993737	0.828601253	4.75E+158	229876715.3
vratio	0.127348643	0.10555586	0.146137787	1.70E+160	6.94E+157
year	0.680584551	0.624217119	0.66388309	3.05E+159	621477120.5
Temperatureinlab	0.9	0.9	0.9	2.91E+162	1.75E+162
ClipboardWeight	0.08559499	0.540709812	0.08559499	4.15E+161	3.47E+161
IIResponse	0.340292276	0.553235908	0.329853862	1.72E+159	1.62E+158
SRConfidenceResponse	0.296450939	0.296450939	0.292275574	2.03E+158	6.36333E+11
NumberofDays	0.9	0.886847599	0.9	1.20E+154	2.40969E+14
Pool2a	0.9	0.898121086	0.9	7.03E+150	20826218.55
Pool2b	0.9	0.892484342	0.9	3.08E+152	183997466.7
Pool2c	0.9	0.896242171	0.9	7.08E+150	2143531.071
Pool2d	0.9	0.898121086	0.9	1.98E+149	1478717.222
Pool3	0.9	0.890605428	0.9	6.83E+150	2.37E+159
Pool4	0.9	0.9	0.9	5.02E+150	71156.24261

Pool5a	0.9	0.892484342	0.9	6.23E+151	304870.1511
Pool6	0.9	0.898121086	0.9	1.94E+151	31347436.82
Pool7b	0.892484342	0.890605428	0.892484342	3.53E+150	19228423.17
Pool7c	0.9	0.894363257	0.9	9.57E+151	105136313.9
Pool7d	0.9	0.898121086	0.9	2.25E+151	234616675.2
Pool8	0.9	0.890605428	0.9	7.75E+151	0.133845584
Pool9	0.9	0.898121086	0.9	9.35E+151	809.7639147
Pool10	0.9	0.898121086	0.9	7.03E+150	20826218.55
Pool11	0.9	0.886847599	0.9	2.43E+152	3.17405E+12
Pool12	0.9	0.884968685	0.9	1.06E+155	5.92E+160
Pool13	0.9	0.884968685	0.9	1.76E+157	4.50E+161
Pool14	0.9	0.886847599	0.9	7.78E+157	4.02E+159
Pool15	0.9	0.88308977	0.9	1.65E+157	1.45E+162
Pool16a	0.9	0.896242171	0.9	5.37E+151	40395491.06
Pool16b	0.9	0.898121086	0.9	1.83E+150	6.21E+156
Pool17	0.9	0.890605428	0.9	1.47E+153	2.50E+159
Pool18	0.9	0.894363257	0.9	2.22E+152	4338.559565
Pool19a	0.9	0.894363257	0.9	7.08E+151	1.95E+157
Pool19b	0.9	0.9	0.9	7.28E+150	135938.0658
Persistence	0.9	0.9	0.9	5.21E+163	3.82E+163
anagrams_order	0.033402923	0.118997912	0.035490605	2.20E+161	3.34E+160
attention_order	0.169102296	0.135699374	0.200417537	2.67E+160	12717195647
availinstruct_order	0.693110647	0.321503132	0.705636743	8.56E+147	59420529.3
availk_order	0.260960334	0.108559499	0.269311065	3.00E+151	1.21E+159
availl_order	0.185803758	0.162839248	0.21085595	5.59E+151	60241735.35
availn_order	0.215031315	0.139874739	0.217118998	8.03E+151	1.62E+159
availr_order	0.223382046	0.144050104	0.225469729	1.40E+150	2.75698E+13
availv_order	0.24217119	0.152400835	0.244258873	3.39E+149	3376054137
bigfive_order	0.187891441	0.137787056	0.175365344	3.03E+158	3.71E+160

debrief_order	0.896242171	0.892484342	0.884968685	9.14E+158	13577950.35
demographics_order	0.888726514	0.894363257	0.886847599	1.36E+158	4818925.778
elmques_order	0.079331942	0.17954071	0.066805846	2.94E+160	8.03E+158
filler1_order	0.9	0.223382046	0.9	3.07E+148	520832.0734
filler2_order	0.9	0.223382046	0.9	3.07E+148	520832.0734
galinskyvignette_order	0.9	0.223382046	0.9	3.07E+148	520832.0734
inlab_order	0.043841336	0.135699374	0.03131524	4.96E+161	3.36E+161
intrinsic_order	0.150313152	0.154488518	0.1565762	4.27E+160	3.87E+157
mcfiller_order	0.9	0.204592902	0.9	7.80E+148	42760221.59
moninvignette_order	0.9	0.204592902	0.9	7.80E+148	42760221.59
mood_order	0.185803758	0.14822547	0.185803758	6.60E+158	3.81832E+14
nfc_order	0.133611691	0.141962422	0.137787056	2.66E+159	3.69087E+14
participantid_order	0.826722338	0.873695198	0.828601253	2.67E+161	1.12E+161
participation_order	0.183716075	0.12526096	0.171189979	1.87E+160	1.20E+158
selfesteem_order	0.152400835	0.135699374	0.158663883	4.82E+160	1.17E+160
startpage_order	0.9	0.9	0.9	0	0
stress_order	0.187891441	0.144050104	0.192066806	6.40E+160	4.14E+160
stroop_order	0.896242171	0.277661795	0.896242171	5.52E+147	10871831.47
stroopinstructions_order	0.898121086	0.277661795	0.9	1.72E+149	10298343.79
stroopinstructionstest_order	0.896242171	0.277661795	0.9	2.38E+147	10390240.21
stroopprac_order	0.898121086	0.277661795	0.9	1.72E+149	10298343.81
tempestimate_order	0.9	0.219206681	0.9	9.18E+148	9322189.043
tempfollowup_order	0.9	0.219206681	0.9	9.18E+148	9322189.043
welcome_order	0.9	0.9	0.9	0	0
MonthComputer	0.9	0.868058455	0.9	1.38E+155	14866823.08
DayComputer	0.817327766	0.096033403	0.824843424	1.71E+150	42962071.8
YearComputer	0.9	0.9	0.9	0	0
DaysSinceMonthComputer	0.894363257	0.868058455	0.898121086	1.27E+151	3.19E+156
DaysSinceAugComputer	0.821085595	0.141962422	0.828601253	5.07E+150	6.33E+157

DaysSinceMonthLab	0.9	0.875574113	0.9	5.00E+159	645663691
DaysSinceAugLab	0.83611691	0.227557411	0.826722338	3.57E+159	5.07E+158
DaysSinceMonthStart	0.9	0.894363257	0.9	2.55E+153	1.65291E+13
DaysSinceAugStart	0.9	0.890605428	0.9	2.28E+153	7260022312
DaysInComp	0.894363257	0.894363257	0.894363257	1.16E+156	94280.5002
DaysInLab	0.898121086	0.898121086	0.898121086	5.20E+156	163953.4713
Openness	0.866179541	0.866179541	0.866179541	7.15E+149	2.43E+159
Conscientiousness	0.864300626	0.864300626	0.864300626	1.03E+150	6.95E+157
Extraversion	0.85302714	0.85302714	0.85302714	3.52E+155	5.90E+158
Agreeableness	0.864300626	0.864300626	0.864300626	1.04E+149	489807749.1
Neuroticism	0.822964509	0.822964509	0.822964509	8.36E+149	1.84E+158
Intrinsic	0.653444676	0.653444676	0.653444676	9.17E+157	64209231.34
Mood	0.858663883	0.858663883	0.858663883	4.83E+158	972914321.9
NFC	0.628392484	0.628392484	0.628392484	1.11E+157	2.43E+157
ReportedAttention	0.898121086	0.804175365	0.886847599	1.36E+149	2.16E+159
ReportedEffort	0.896242171	0.832359081	0.888726514	1.52E+150	4.13E+159
SelfEsteem	0.9	0.498956159	0.86993737	1.11E+148	21664074.41
Stress	0.699373695	0.699373695	0.699373695	6.18E+157	2.52E+159
K1st	0.9	0.841753653	0.871816284	7.77E+154	170490195.2
L1st	0.9	0.849269311	0.875574113	5.44E+158	201427563
N1st	0.9	0.847390397	0.881210856	1.14E+158	203680471
R1st	0.9	0.837995825	0.881210856	2.32E+157	285661514.3
V1st	0.9	0.847390397	0.873695198	4.27E+158	255903953
AvailFirst	0.776617954	0.780793319	0.764091858	1.83E+158	3.35E+157
ArgumentQuality	0.511482255	0.511482255	0.511482255	4.43E+158	1.13E+159
NFCcenter	0.832359081	0.832359081	0.832359081	2.68E+156	1317284341
ELMCond	0	0.572025052	0	2.12E+160	7.97E+159
CBReject	0.802296451	0.802296451	0.802296451	7.08E+157	11627661.86

Conclusion:

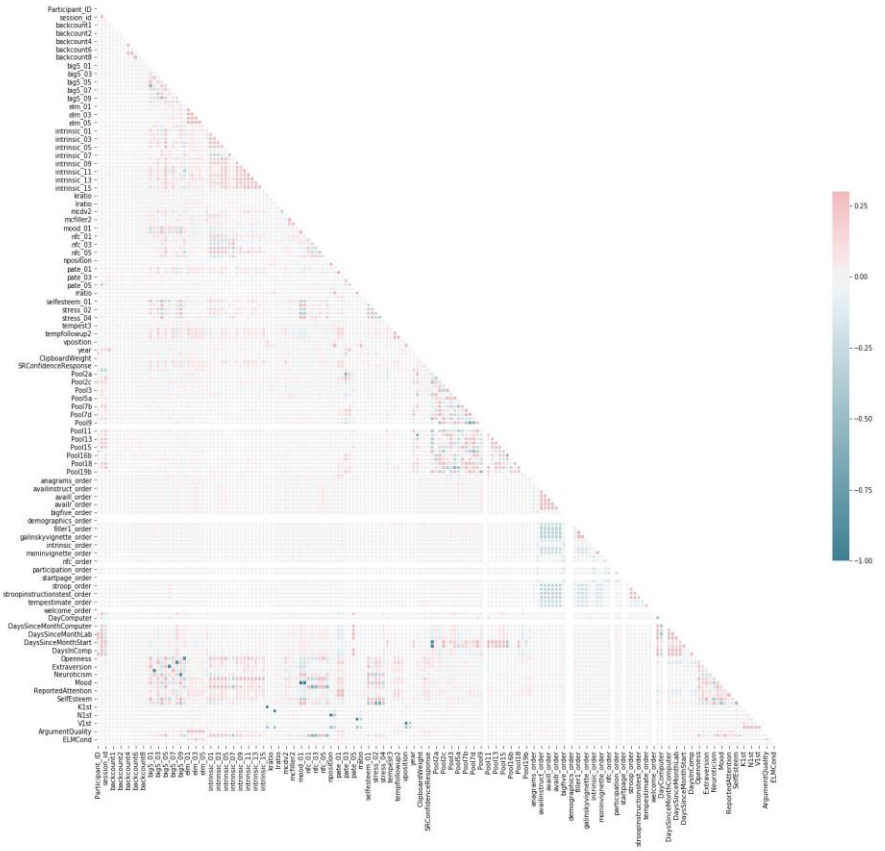
Based on the accuracy of our models, we chosed Random Forest Regression as our final model for prediction.

Question 4 Evaluate your Model: Where is your model particularly successful, where does it lack? Does it need a certain amount of features in order to interpolate well? Are there some features it is really good at predicting and some it is really poor at predicting? Why do you think that is?

Our model was working great for the classification algorithms as compared to regression based algorithms. We did some further analysis to extract important features as follows:

Coorelation Matrix

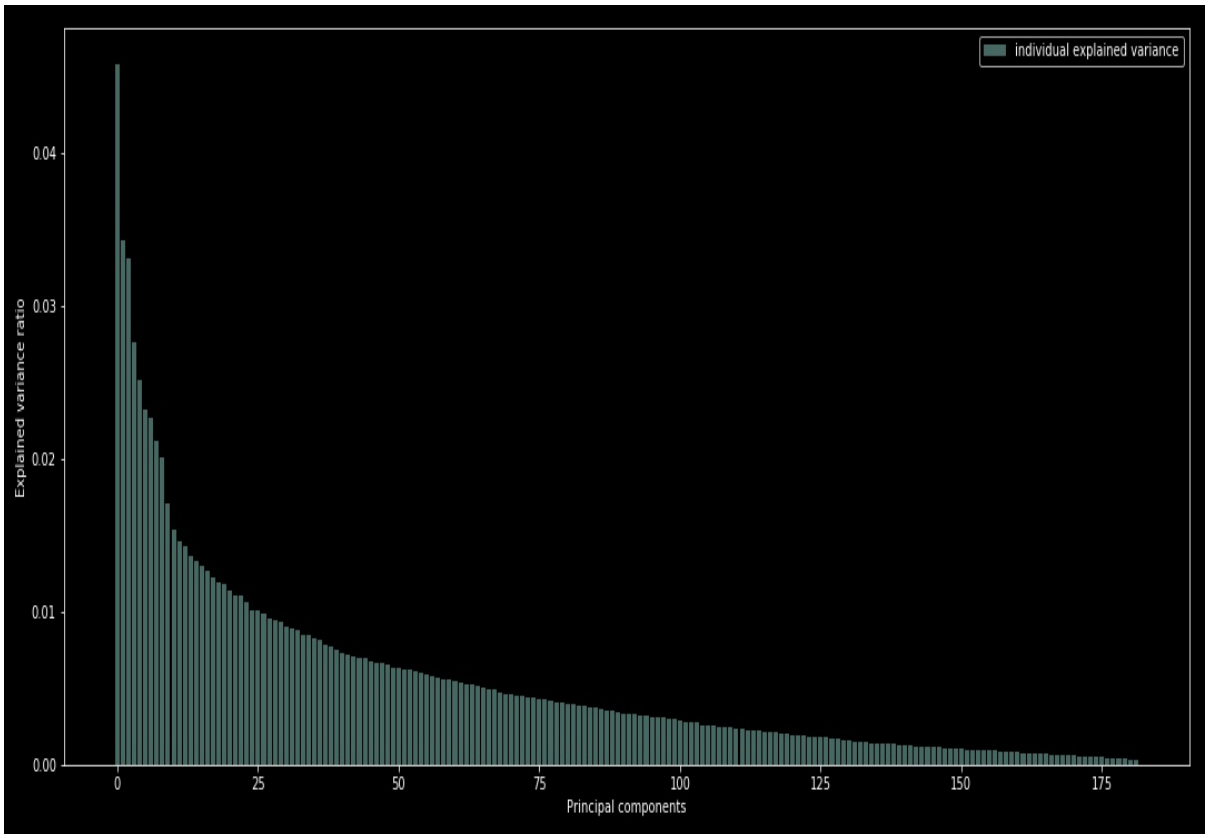
We calculated the coorelation matrix between our all features inorder to check the dependency and how our features are related to each other. A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used as a way to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.



Some of the features had very strong inverse coorelation while most of the features had very weak coorelation between them, so we didn't got any useful insights from them.

PCA:

We performed the PCA on our feature set and tried to identify the key features. The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.



We identified some key features from our dataset which have a higher value of variance.

imp_feature_list = ['Participant_ID', 'RowNumber', 'session_id', 'age', 'backcount1', 'backcount10', 'backcount2', 'backcount3', 'backcount4', 'backcount5', 'backcount6', 'backcount7', 'backcount8', 'backcount9', 'big5_01', 'big5_02', 'big5_03', 'big5_04', 'big5_05', 'big5_06', 'big5_07', 'big5_08', 'big5_09', 'big5_10', 'elm_01', 'elm_02', 'elm_03', 'elm_04', 'elm_05', 'gender', 'intrinsic_01', 'intrinsic_02', 'intrinsic_03', 'intrinsic_04', 'intrinsic_05', 'intrinsic_06', 'intrinsic_07', 'intrinsic_08', 'intrinsic_09', 'intrinsic_10', 'intrinsic_11', 'intrinsic_12', 'intrinsic_13', 'intrinsic_14', 'intrinsic_15', 'kposition', 'kratio', 'lposition', 'lratio', 'mcdv1', 'mcdv2', 'mcfiller1', 'mcfiller2', 'mcfiller3', 'mood_01', 'mood_02', 'nfc_01', 'nfc_02', 'nfc_03', 'nfc_04', 'nfc_05', 'nfc_06', 'nposition', 'nratio', 'pate_01', 'pate_02', 'pate_03', 'pate_04', 'pate_05', 'rposition', 'rratio', 'sarcasm', 'selfesteem_01', 'stress_01', 'stress_02', 'stress_03', 'stress_04', 'tempest2', 'tempest3', 'tempfollowup1', 'tempfollowup2', 'tempfollowup3', 'vposition', 'vratio', 'year', 'Temperatureinlab', 'ClipboardWeight', 'IIResponse',

'SRConfidenceResponse', 'NumberofDays', 'Pool2a', 'Pool2b', 'Pool2c', 'Pool2d', 'Pool3', 'Pool4', 'Pool5a', 'Pool6', 'Pool7b', 'Pool7c', 'Pool7d', 'Pool8', 'Pool9', 'Pool10', 'Pool11', 'Pool12', 'Pool13', 'Pool14', 'Pool15', 'Pool16a', 'Pool16b', 'Pool17', 'Pool18', 'Pool19a', 'Pool19b', 'Persistence', 'anagrams_order', 'attention_order', 'availinstruct_order', 'availk_order', 'availl_order', 'availn_order', 'availr_order', 'availv_order', 'bigfive_order']

Apart from the above mentioned features, the other features have comparitively low variance value and it was getting decreased as we down from left to right in the PCA chart.

['debrief_order', 'demographics_order', 'elmques_order', 'filler1_order', 'filler2_order', 'galinskyvignette_order', 'inlab_order', 'intrinsic_order', 'mcfiller_order', 'moninvignette_order', 'mood_order', 'nfc_order', 'participantid_order', 'participation_order', 'selfesteem_order', 'startpage_order', 'stress_order', 'stroop_order', 'stroopinstructions_order', 'stroopinstructionstest_order', 'stroopprac_order', 'tempestimate_order', 'tempfollowup_order', 'welcome_order', 'MonthComputer', 'DayComputer', 'YearComputer', 'DaysSinceMonthComputer', 'DaysSinceAugComputer', 'DaysSinceMonthLab', 'DaysSinceAugLab', 'DaysSinceMonthStart', 'DaysSinceAugStart', 'DaysInComp', 'DaysInLab', 'Openness', 'Conscientiousness', 'Extraversion', 'Agreeableness', 'Neuroticism', 'Intrinsic', 'Mood', 'NFC', 'ReportedAttention', 'ReportedEffort', 'SelfEsteem', 'Stress', 'K1st', 'L1st', 'N1st', 'R1st', 'V1st', 'AvailFirst', 'ArgumentQuality', 'NFCcenter', 'ELMCond', 'CBReject']

We compared our model prediction by considering all the features together against the PCA specific features in our training dataset as the dependent feature for target feature. We found that our model is predicting substantially less with the high variance features as compared to all the features taken together for prediction.

If we will just consider the PCA generated features, then our model is tend to predict values with less accuracy of 5-7 percent. Basically there is a trade off between the computation power and model accuracy. Considering the 125 PCA features will run the model in less computation time and will give less accuracy as compared to model with 180 plus features with high computation time and high accuracy.

Question 5: Analyze the Data: What features were particularly valuable in predicting/interpolating? What features weren't particularly useful at all? Were there any features about the researcher/experimental environment that were particularly relevant to predicting answers - and if so, what conclusions can you draw about the replicability of those effects?

As mentioned in Question 4, we extracted the important features using the PCA and coorelation matrix.

Features which were particularly valuable in predicting/interpolating are

imp_feature_list = ['Participant_ID', 'RowNumber', 'session_id', 'age', 'backcount1', 'backcount10', 'backcount2', 'backcount3', 'backcount4', 'backcount5', 'backcount6', 'backcount7', 'backcount8', 'backcount9', 'big5_01', 'big5_02', 'big5_03', 'big5_04', 'big5_05', 'big5_06', 'big5_07', 'big5_08',

'big5_09', 'big5_10', 'elm_01', 'elm_02', 'elm_03', 'elm_04', 'elm_05', 'gender', 'intrinsic_01', 'intrinsic_02', 'intrinsic_03', 'intrinsic_04', 'intrinsic_05', 'intrinsic_06', 'intrinsic_07', 'intrinsic_08', 'intrinsic_09', 'intrinsic_10', 'intrinsic_11', 'intrinsic_12', 'intrinsic_13', 'intrinsic_14', 'intrinsic_15', 'kposition', 'kratio', 'lposition', 'lratio', 'mcdv1', 'mcdv2', 'mcfiller1', 'mcfiller2', 'mcfiller3', 'mood_01', 'mood_02', 'nfc_01', 'nfc_02', 'nfc_03', 'nfc_04', 'nfc_05', 'nfc_06', 'nposition', 'nratio', 'pate_01', 'pate_02', 'pate_03', 'pate_04', 'pate_05', 'rposition', 'rratio', 'sarcasm', 'selfesteem_01', 'stress_01', 'stress_02', 'stress_03', 'stress_04', 'tempest2', 'tempest3', 'tempfollowup1', 'tempfollowup2', 'tempfollowup3', 'vposition', 'vratio', 'year', 'Temperatureinlab', 'ClipboardWeight', 'IIResponse', 'SRConfidenceResponse', 'NumberofDays', 'Pool2a', 'Pool2b', 'Pool2c', 'Pool2d', 'Pool3', 'Pool4', 'Pool5a', 'Pool6', 'Pool7b', 'Pool7c', 'Pool7d', 'Pool8', 'Pool9', 'Pool10', 'Pool11', 'Pool12', 'Pool13', 'Pool14', 'Pool15', 'Pool16a', 'Pool16b', 'Pool17', 'Pool18', 'Pool19a', 'Pool19b', 'Persistence', 'anagrams_order', 'attention_order', 'availinstruct_order', 'availk_order', 'availl_order', 'availn_order', 'availr_order', 'availv_order', 'bigfive_order']

Features weren't particularly useful at all :

Since, we need to train the data over our model and we need to make sure that we chose appropriate features as the input, at first, we removed all the features where more than 50 percent of the data was missing only if that variable is insignificant.

{'SomeEndorse','MostEndorse','worstgrade_order','tempcomm_order','tempcomf_order','tempagenm_order','tempagenf_order','mcsome_order','mcmost_order','lowpower_order','highpower_order','elmweak_order','elmstrong_order','consent_order','compcode_order','bestgrade_order','Pool20','Pool16c','Pool7e','Pool7a','Pool5b','Pool2e','Pool1','Notes','SRTFCorrect','worstgrade5','worstgrade4','worstgrade3,'worstgrade2','worstgrade1','anagrams1', 'anagrams2', 'anagrams3', 'anagrams4', 'attention', 'attentioncorrect', 'bestgrade1', 'bestgrade2','bestgrade3', 'bestgrade4', 'bestgrade5', 'lowpower', 'mcmost1', 'mcmost2', 'mcmost3', 'mcmost4', 'mcmost5','mcsome1', 'mcsome2', 'mcsome3', 'mcsome4', 'mcsome5'}

which reduced our features to 210 features total.

Secondly, since almost 98 percent of the data was numeric data and we were facing many problems if data was in String format as we were not able to convert it into float, so we need to remove all the features from our dataset where String data was used..

drop_col = ['ethnicity', 'Site', 'major', 'tempest1', 'Date.x', 'Experimenter', 'OrderofTasks', 'SRCondition', 'SRMeetingResponse', 'StartDate.x', 'EndDate', 'ClipBoardMaterial', 'DateComputer', 'TimeComputer', 'AttentionCheck', 'PowerCond', 'CredCond', 'Genderfactor', 'SubDistCond', 'TempCond4', 'TempCond', 'TargetGender']

#df = df.drop(columns=drop_col)

.

The original data consisted of 274 columns. After preprocessing (method mentioned above) we brought it down to 182 columns. On this preprocessed data we used dimensionality reduction techniques such as PCA and correlation chart to view which features were highly valuable while predicting.

The missing value depends on the hypothetical value (e.g. People with high salaries generally do not want to reveal their incomes in surveys) or missing value is dependent on some other

variable's value (e.g. Let's assume that females generally don't want to reveal their ages! Here the missing value in age variable is impacted by gender variable)

In our case, if the features like age and gender tends to fall in the PCA extracted feature set then our model tends to give a higher accuracy value for the prediction made but if these features are not present in the PCA feature training set then our model simply doesn't get any insights from it and tend to give the prediction on the overall basis.

Question 6: Generate Data: Use your system to try to generate realistic data, and compare your generated data to the real data. How good does it look? What does it mean for it to `look good'?

We have taken two approaches to generate new data, explained below:

1. Approach I:
    a. Taking a subset of seen rows (25 % of our data set) and putting NA at random 10% of rows. Now using the precomputed models above to impute these values. Thus, generating new data.
    b. We use all of this data for model creation using Random Forest model, and compute accuracy score. If the accuracy varies by 5% from the original, we say that our data looks good. Looking good here means that the data follows the same distribution and gives predictions similar to the original data.
    c. We can quantify this based on our observation in the table below. Where we present the accuracy score for each feature using synthetic data.
    d. The prediction accuracy on the synthetic data is around 80% of the original data. We can see that in the chart below.
2. Approach 2:
    a. Taking a subset of seen rows (25 % of our data set) and putting NA at random 10% of rows and finally adding random noise using Uniform distribution to remaining rows. Now using the precomputed models above to impute these values. Thus, generating new data.
    b. We use all of this data for model creation using Random Forest model, and compute accuracy score. If the accuracy varies by 5% from the original, we say that our data looks good. Looking good here means that the data follows the same distribution and gives predictions similar to the original data.
    c. We can quantify this based on our observation in the table below. Where we present the accuracy score for each feature using synthetic data.
    d. The prediction accuracy on the synthetic data is around 80% of the original data. We can see that in chart below.

Feautures	Approach 1	Approach 2	Original
Participant_ID	0.008333333	0	0.131524008
RowNumber	0.008333333	0	0.146137787
session_id	0	0	0.008350731

age	0.183333333	0.091666667	0.565762004
backcount1	0.066666667	0.066666667	0
backcount10	0.058333333	0.058333333	0.549060543
backcount2	0.066666667	0.2	0.736534447
backcount3	0.15	0.133333333	0.75908142
backcount4	0.216666667	0.083333333	0.797494781
backcount5	0.175	0.108333333	0.744050104
backcount6	0.158333333	0.091666667	0.775991649
backcount7	0.058333333	0.083333333	0.797494781
backcount8	0.15	0.058333333	0.777870564
backcount9	0.025	0.041666667	0.792901879
big5_01	0.35	0.308333333	0.856784969
big5_02	0.325	0.25	0.839874739
big5_03	0.458333333	0.391666667	0.837995825
big5_04	0.308333333	0.441666667	0.817327766
big5_05	0.416666667	0.275	0.837995825
big5_06	0.35	0.216666667	0.807933194
big5_07	0.408333333	0.383333333	0.856784969
big5_08	0.358333333	0.258333333	0.839874739
big5_09	0.25	0.325	0.832359081
big5_10	0.275	0.308333333	0.843632568
elm_01	0.275	0.25	0.329853862
elm_02	0.266666667	0.208333333	0.388308977
elm_03	0.225	0.266666667	0.407098121
elm_04	0.308333333	0.208333333	0.409185804
elm_05	0.333333333	0.275	0.356993737
gender	0.75	0.675	0.711899791
intrinsic_01	0.458333333	0.491666667	0.528183716
intrinsic_02	0.45	0.458333333	0.509394572
intrinsic_03	0.483333333	0.5	0.524008351
intrinsic_04	0.483333333	0.483333333	0.409185804
intrinsic_05	0.383333333	0.383333333	0.457202505
intrinsic_06	0.483333333	0.5	0.463465553
intrinsic_07	0.508333333	0.416666667	0.455114823
intrinsic_08	0.433333333	0.491666667	0.417536534
intrinsic_09	0.325	0.383333333	0.400835073
intrinsic_10	0.416666667	0.291666667	0.342379958
intrinsic_11	0.358333333	0.358333333	0.386221294
intrinsic_12	0.425	0.475	0.419624217
intrinsic_13	0.391666667	0.375	0.350730689
intrinsic_14	0.341666667	0.441666667	0.352818372
intrinsic_15	0.425	0.45	0.43006263

kposition	0.7	0.7	0.866179541
kratio	0.041666667	0.066666667	0.131524008
lposition	0.65	0.633333333	0.85302714
lratio	0.016666667	0.041666667	0.152400835
mcdv1	0.65	0.616666667	0.682672234
mcdv2	0.133333333	0.191666667	0.223382046
mcfiller1	0.533333333	0.583333333	0.563674322
mcfiller2	0.608333333	0.641666667	0.590814196
mcfiller3	0.633333333	0.658333333	0.699373695
mood_01	0.241666667	0.291666667	0.819206681
mood_02	0.291666667	0.208333333	0.839874739
nfc_01	0.366666667	0.325	0.363256785
nfc_02	0.408333333	0.466666667	0.507306889
nfc_03	0.325	0.358333333	0.344467641
nfc_04	0.383333333	0.516666667	0.415448852
nfc_05	0.358333333	0.35	0.4217119
nfc_06	0.45	0.358333333	0.423799582
nposition	0.475	0.508333333	0.832359081
nratio	0.058333333	0.041666667	0.129436326
pate_01	0.658333333	0.7	0.898121086
pate_02	0.633333333	0.658333333	0.896242171
pate_03	0.866666667	0.825	0.791022965
pate_04	0.708333333	0.641666667	0.729018789
pate_05	0.25	0.241666667	0.331941545
rposition	0.491666667	0.566666667	0.826722338
rratio	0.025	0.025	0.9
sarcasm	0.166666667	0.166666667	0.152400835
selfesteem_01	0.608333333	0.583333333	0.9
stress_01	0.433333333	0.475	0.519832985
stress_02	0.316666667	0.35	0.530271399
stress_03	0.291666667	0.341666667	0.475991649
stress_04	0.375	0.425	0.668058455
tempest2	0.258333333	0.275	0.263048017
tempest3	0.383333333	0.416666667	0.319415449
tempfollowup1	0.341666667	0.341666667	0.352818372
tempfollowup2	0.375	0.341666667	0.315240084
tempfollowup3	0.316666667	0.325	0.286012526
vposition	0.641666667	0.6	0.828601253
vratio	0.016666667	0.016666667	0.146137787
year	0.591666667	0.516666667	0.66388309
Temperatureinlab	0.033333333	0.066666667	0.9
ClipboardWeight	0.05	0.041666667	0.08559499

IIResponse	0.25	0.266666667	0.329853862
SRConfidenceResponse	0.266666667	0.283333333	0.292275574
NumberofDays	0.258333333	0.191666667	0.9
Pool2a	0.883333333	0.941666667	0.9
Pool2b	0.6	0.6	0.9
Pool2c	0.883333333	0.8	0.9
Pool2d	0.925	0.925	0.9
Pool3	0.266666667	0.283333333	0.9
Pool4	0.966666667	0.95	0.9
Pool5a	0.916666667	0.916666667	0.9
Pool6	0.958333333	0.941666667	0.9
Pool7b	0.791666667	0.825	0.892484342
Pool7c	0.875	0.841666667	0.9
Pool7d	0.891666667	0.916666667	0.9
Pool8	0.75	0.8	0.9
Pool9	0.941666667	0.958333333	0.9
Pool10	1	1	0.9
Pool11	0.6	0.558333333	0.9
Pool12	0.058333333	0.15	0.9
Pool13	0.066666667	0.016666667	0.9
Pool14	0.041666667	0.083333333	0.9
Pool15	0.15	0.175	0.9
Pool16a	0.891666667	0.966666667	0.9
Pool16b	0.925	0.841666667	0.9
Pool17	0.491666667	0.491666667	0.9
Pool18	0.9	0.833333333	0.9
Pool19a	0.483333333	0.408333333	0.9
Pool19b	1	1	0.9
Persistence	0.025	0	0.9
anagrams_order	0.041666667	0.033333333	0.035490605
attention_order	0.116666667	0.083333333	0.200417537
availinstruct_order	0.225	0.325	0.705636743
availk_order	0.191666667	0.108333333	0.269311065
availl_order	0.116666667	0.2	0.21085595
availn_order	0.116666667	0.15	0.217118998
availr_order	0.225	0.166666667	0.225469729
availv_order	0.183333333	0.158333333	0.244258873
bigfive_order	0.133333333	0.108333333	0.175365344
debrief_order	1	1	0.884968685
demographics_order	0.125	0.141666667	0.886847599
elmques_order	0.041666667	0.041666667	0.066805846
filler1_order	0.416666667	0.425	0.9

filler2_order	0.433333333	0.333333333	0.9
galinskyvignette_order	0.433333333	0.291666667	0.9
inlab_order	0.033333333	0.091666667	0.03131524
intrinsic_order	0.125	0.116666667	0.1565762
mcfiller_order	0.208333333	0.325	0.9
moninvignette_order	0.208333333	0.175	0.9
mood_order	0.191666667	0.175	0.185803758
nfc_order	0.083333333	0.058333333	0.137787056
participantid_order	1	1	0.828601253
participation_order	0.066666667	0.175	0.171189979
selfesteem_order	0.183333333	0.116666667	0.158663883
startpage_order	1	1	0.9
stress_order	0.116666667	0.191666667	0.192066806
stroop_order	0.375	0.508333333	0.896242171
stroopinstructions_order	0.458333333	0.325	0.9
stroopinstructionstest_order	0.358333333	0.358333333	0.9
stroopprac_order	0.408333333	0.358333333	0.9
tempestimate_order	0.3	0.191666667	0.9
tempfollowup_order	0.316666667	0.166666667	0.9
welcome_order	1	1	0.9
MonthComputer	0.683333333	0.8	0.9
DayComputer	0.116666667	0.15	0.824843424
YearComputer	1	1	0.9
DaysSinceMonthComputer	0.175	0.25	0.898121086
DaysSinceAugComputer	0.058333333	0.108333333	0.828601253
DaysSinceMonthLab	0.066666667	0.175	0.9
DaysSinceAugLab	0.108333333	0.058333333	0.826722338
DaysSinceMonthStart	0.275	0.383333333	0.9
DaysSinceAugStart	0.3	0.391666667	0.9
DaysInComp	1	0.8	0.894363257
DaysInLab	0.983333333	0.708333333	0.898121086
Openness	0.416666667	0.283333333	0.866179541
Conscientiousness	0.258333333	0.241666667	0.864300626
Extraversion	0.45	0.308333333	0.85302714
Agreeableness	0.425	0.333333333	0.864300626
Neuroticism	0.391666667	0.4	0.822964509
Intrinsic	0.616666667	0.458333333	0.653444676
Mood	0.241666667	0.133333333	0.858663883
NFC	0.466666667	0.491666667	0.628392484
ReportedAttention	0.691666667	0.691666667	0.886847599
ReportedEffort	0.625	0.658333333	0.888726514
SelfEsteem	0.658333333	0.5	0.86993737

Stress	0.433333333	0.483333333	0.699373695
K1st	0.525	0.541666667	0.871816284
L1st	0.475	0.558333333	0.875574113
N1st	0.616666667	0.633333333	0.881210856
R1st	0.616666667	0.591666667	0.881210856
V1st	0.575	0.583333333	0.873695198
AvailFirst	0.491666667	0.466666667	0.764091858
ArgumentQuality	0.416666667	0.466666667	0.764091858
NFCcenter	0.933333333	0.933333333	0.511482255
ELMCond	0.141666667	0.625	0
CBReject	0.891666667	0.891666667	0.802296451

Bonus Question

Question 1: Build your system to include processing and analysis of natural language answers, to try to use them to inform prediction of other features. How can you represent natural language answers in a useful way? What language processing is necessary to be able to use them? Do they actually provide useful information for the prediction problem?

We identified columns (feedback, attentioncorrect) where we have to use Natural Language Processing. We then did the Natural Language Processing in following steps:

Steps:

1. We decided to use Bag of words model for the above column as the sentences are relatively small and use a small word vocabulary.
2. Now to use this model we need to strip each sentence in to their respective words.
3. Remove any punctuation and stop words from the above list.
4. Now we replace the words with their root value.
5. We create the word frequency dictionary. Example = {‘read’:100, ‘instruction’:100}
6. In the data we replace sentences with words vectors created using above model. So if the sentence is: “I read the instructions” it will be replaced with [1, 1] vector.
7. We use this vector when training the models in question 2.

After doing above the process we saw that most of the values are same, remaining 50% of the rows were missing. As the values were the same, we used mode method to fill the column.

But using this feature had no improvement on the model so we discarded these columns. Hence, we say that these features were not useful for your prediction problem.

Thank You Note:

Thanks a lot Professor!! For all the learning and knowledge sharing session during the lectures. We are the newbies in machine learning and we learnt a great deal of substantial knowledge regarding the various models from the data scientist as well as mathematician prospect.