CS 536: Machine Learning **Professor: Charles Cowan**

Final Project - Data Completion and Interpolation

Name: Keya Desai*, NetID:* kd706

Name: Prakruti Joshi*, NetID:* phj15

In this project, we take a dataset of the form $\{\underline{x}^1, \underline{x}^2, \ldots, \underline{x}^m\}$ and construct a system for predicting or interpolating missing features. In this case, instead of singling out a single feature or factor for prediction/regression/classification, any feature of X might be missing and need to be predicted or reconstructed from the features that are present - and the features that are present may vary from instance to instance.

1 Data

The data that we have used for the project is the Adult dataset or the Census Income data. The data is extracted from the 1994 Census database and aims to predict whether income of a person exceeds $50K/year based on information of age, gender, education, race, work class, and demographics. The data has approximately 48K such instances with 14 attributes - 6 continuous and 8 categorical. Numerical attributes are - age, fnlwgt (a number calculated to several factors of demographics), educationNum, capitalGain, capitalLoss and hoursPerWeek. Categorical attributes are - workclass, educationClass, maritalStatus, occupation, relationship, race, sex, nativeCountry and income (>=$50K/yr or <$50K/yr).

The code for this project can be found on Github. The task has been performed in Python using the libraries of numpy, pandas and matplotlib with the models coded from scratch.

1.1 Data Representation

Before starting with modelling, it is necessary to understand the data and clean it. The numerical and categorical features needs to be handled separately.

- Numerical data - The range of different numerical features varies largely - for example *age* has a range of [19, 70] where as *capital Gain* has a range of [0, 99999]. Normalising the data is a good practice to bring all the numerical features in a range of [0, 1]. Since underlying distribution of the data is not known, min-max normalisation is employed -

$$X` = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Categorical data - Since the input to most algorithms is numerical data, the categorical data is converted to numerical data using one-hot encoding. For example - the column of *sex* has two unique values - *Female and Male*. These two classes are converted to two separate columns of *Is_sex_female* and *Is_sex_male*, with values being either [1, 0] or [0, 1] for each instance. Similar process has been followed for all categorical attributes, increasing the number of features from 15 to 106.
- Missing data - The missing data in the data is represented using " ?". There are 1221 rows total in training and testing data with atleast one feature missing. Such rows can not be used in training, testing or validation and hence are removed to form a separate dataset that can be treated as real world data for testing.
- Reindexing of test data -
  - The one-hot encoded feature columns are formed in the order they appear in the data. For instance - if *female* appears before *male* in the train data then the order of the columns will be *Is_sex_female* and *Is_sex_male*. If in train data *male* appears before *male* then the column order will be reversed. Hence it is important to reorder the columns in test data according to the order in train data since both the pandas dataframes are converted to numpy for passing to algorithms.

Feature name	**Number of Categories**	**After pruning**
work Class	7	6
education	16	16
maritalStatus	7	6
occupation	14	13
relationship	6	6
race	5	5
gender	2	2
nativeCountry	41	24
income	2	2
Total categorical features	98	80

Table 1: Count of categorical feature classes and count of categorical features after pruning classes with less data representation

  - No instance with country *"Holland-Netherlands"* appears in the test data and hence the column *Is_nativeCountry_Holland-Netherlands* needs to be added with all values as 0.

- Missing data generation - Missing data is to be generated from the entire dataset so that it can be used for training the models. A *(m x k)* array *isFeatureReal* is used to store whether the feature value in training/testing data is real(1) or missing(0). The algorithm for generating missing data is as follows - For each row of the data -
  - Select number of features to remove from $num\_missing \sim Uniform(0, num\_missing\_max)$ where num_missing_max is taken as 3.
  - Randomly select *num_missing* features to remove from original 15 features.
  - If the feature to be removed is numerical, it can be directly set to NaN/0 depending on the model. The corresponding element in *IsFeatureReal* is also set to 0.
  - If the feature is categorical, then all the corresponding one-hot encoded columns also need to be assigned a value of NaN/0 along with setting all the elements in *isFeatureReal* to 0.

  On average, 322K missing data points are produced for a matrix of size (30162 x 106) i.e approximately 10% of the data is missing in data.

1.2 Data Analysis and Feature selection

To understand the distribution of data and the correlation between features, we performed data analysis on the features. The data analysis give a fair idea about the importance of the feature overall.

- Since we have many categorical features with many classes, the first thing that we did was to analyse the distribution of the categorical classes. It can be seen that the distribution of classes is not uniform for most features (1) suggesting imbalance in data. Categories with less than 30 instances are prune to reduce the feature space. The reduction in number of classes per feature can be seen in 1.

- The column of educationNum is encoding of education column. For instance - Bachelors is encoded as 13, Class 11 is encoded as 7 etc. Hence educationNum is a redundant feature and can be removed while training the model.

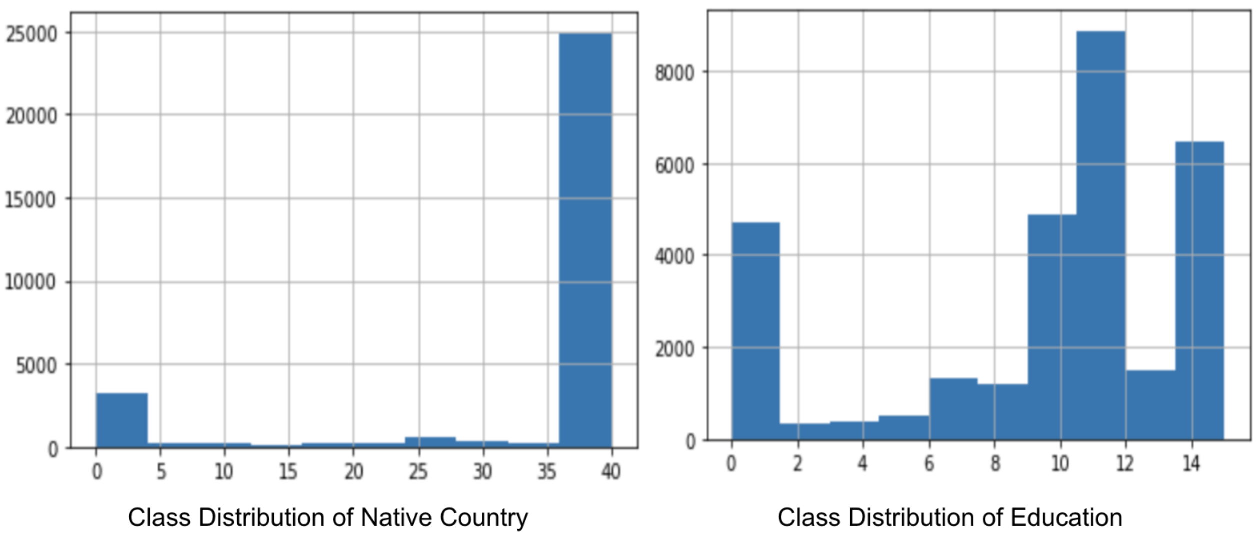
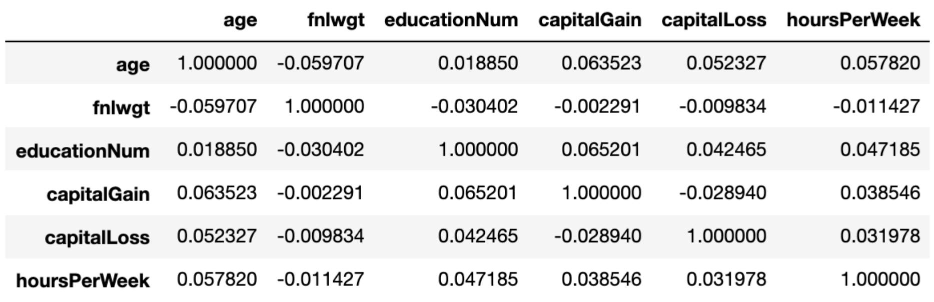


Figure 1: Class distribution of Categorical variables: Native Country and Education

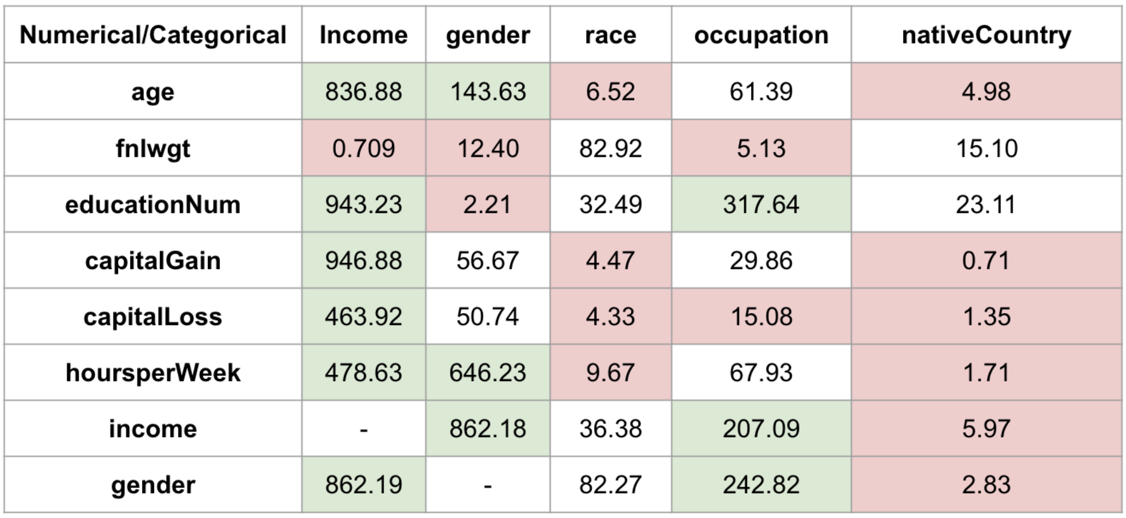
- Correlation between numerical data - Correlation between numerical data is easier to find and hence first we analyse correlation between numerical features to observe the dependencies. We can see the positive and the negative correlation in 2. The correlation coefficient lies between +1 and -1 indicating strong linear relationship and 0 indicates no linear relation. The values in the table suggest that there is little linear relationship between the numerical features.



	age	fnlwgt	educationNum	capitalGain	capitalLoss	hoursPerWeek
age	1.000000	-0.059707	0.018850	0.063523	0.052327	0.057820
fnlwgt	-0.059707	1.000000	-0.030402	-0.002291	-0.009834	-0.011427
educationNum	0.018850	-0.030402	1.000000	0.065201	0.042465	0.047185
capitalGain	0.063523	-0.002291	0.065201	1.000000	-0.028940	0.038546
capitalLoss	0.052327	-0.009834	0.042465	-0.028940	1.000000	0.031978
hoursPerWeek	0.057820	-0.011427	0.047185	0.038546	0.031978	1.000000

Figure 2: Correlation between numerical features

- Correlation between categorical and numerical features - The ANOVA test assesses whether the averages of more than two groups are statistically different from each other. This analysis is appropriate for comparing the averages of a numerical variable for more than two categories of a categorical variable. The value of the f-statistic determines the acceptance/rejection of the null hypothesis of the features being dependent. The higher the f-statistic, more dependent are the features.



Numerical/Categorical	Income	gender	race	occupation	nativeCountry
age	836.88	143.63	6.52	61.39	4.98
fnlwgt	0.709	12.40	82.92	5.13	15.10
educationNum	943.23	2.21	32.49	317.64	23.11
capitalGain	946.88	56.67	4.47	29.86	0.71
capitalLoss	463.92	50.74	4.33	15.08	1.35
hoursperWeek	478.63	646.23	9.67	67.93	1.71
income	-	862.18	36.38	207.09	5.97
gender	862.19	-	82.27	242.82	2.83

Figure 3: Anova test f-statistic to analyse dependency between categorical and numerical feature. The higher values of F-statistic indicate dependency between features and is represented in green. The lower values of F-statistic indicate lower dependency between features and is represented in red.

From the above table 3, we can see that income is highly correlated to features such as education, age, gender (not that surprising since the data is skewed towards Males and was collected in 1994 Census!) Since the demographics determines features like income, capitalGain, Loss, we expected it to have stronger correlations. We reasoned this might be due the large number of classes in Native Country reducing overall records per class.

- The correlation analysis gives us an overview of the important features while predicting other features. Furthermore, a chi-squared test between the categorical variables would have given more insights into the inter-dependencies. However, due to time constraints we were not able to implement that. Overall, Income and gender are one of the most important features in the data.

1.3 Problem of Interpolation

- The problem of interpolation that we are considering over here is that of missing features in our data. We want a model that takes in the data entry containing non-missing and the missing features and returns an output vector which fills in the missing values. We are looking at a way of predicting the missing features given the value of other known features.
- To predict or interpolate a missing feature means that the model should be able to impute the value of the feature based on the acceptable range of values of the feature. The model should also consider the underlying dependencies between features in the data.
- If we assume that the data is based on a linear distribution or any other distribution, then problem of interpolation can be thought of as predicting the value of the feature that maximizes the likelihood of the feature given other variables.

2 Model

The next step after exploring and preparing the data is modeling. The models implemented for the system for predicting/interpolating missing features are described in the subsection.

2.1 Model description

2.1.1 Baseline

The basic completion agent requires filling the missing values as the mean for numerical feature mode for categorical feature. Thus, the prediction of a particular feature is independent of other features in the data point. Thus, this model ensures that the completed data point will then have the 'typical' properties of the data set. This forms a good baseline model to compare the performance of the other models.

2.1.2 K-Nearest Neighbors

This method is an extension of the baseline model where we introduce the influence of other features for predicting the current feature by inculcating similarity between data points. The similarity between the two features is computed by using the Euclidean distance between the features. The lesser the distance, more similar are the data points.

In this model, the number of nearest neighbors (k) is modeled as a hyper-parameter.k-nearest data points are selected based on the prec-computed similarity scores. Here we use the sklearn's pairwise euclidean similarity function for faster computation speed. The mean/mode of the feature is predicted based on the value of K data points. It can be seen in Figure 4 that the testing error of the model decreases as the number of neighbours k is increases.

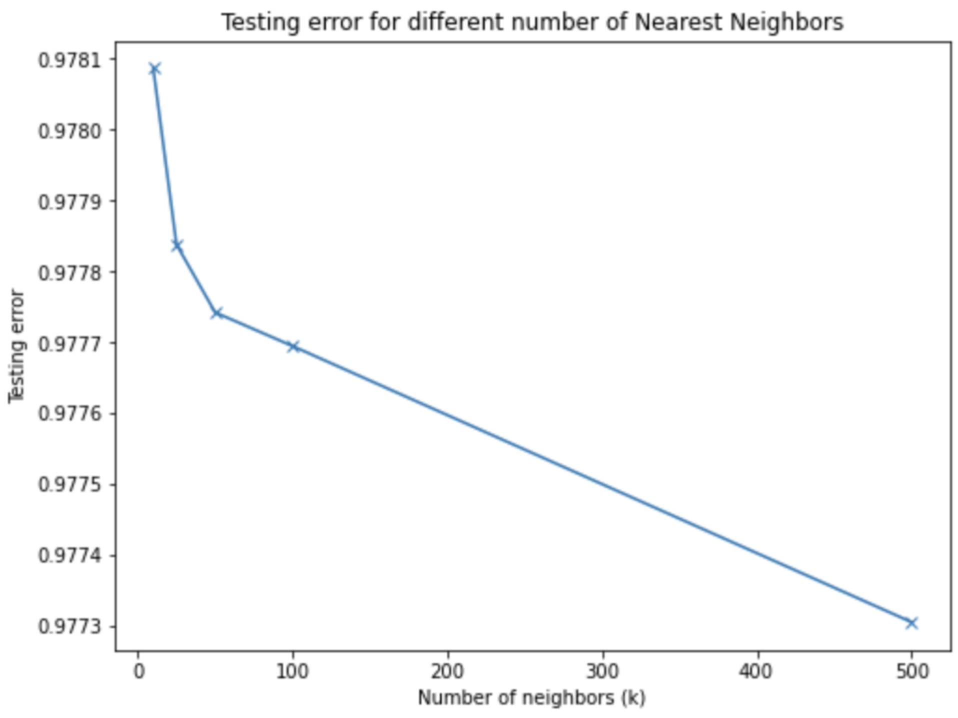


Figure 4: Testing RMSE of knn model for varying k values.

Hence, this model is able to include other features while predicting the current feature value. There is a slight computation overhead of computing similarity for all points across all the data points available. However, this model still relies on the wisdom of the crowd and fails to generalize relationship between features.

2.1.3 Sequential Prediction

The baseline model described above does not take into account any of the features of x that are actually present directly. KNN captures the relationship among features to some extent. But we can design a model that learns from these relationships, that can be used to predict missing data. This modifies our input space into a few features which are not missing and output space into a single feature that we are trying to predict. The goal for the regression model would be to learn weights that captures the dependence among the input features and the output feature.

Suppose there are 3 features x_1, x_2 and x_3. We need to formulate a model which finds interpolation among all these 3 features, which can be achieved using Linear Regression. For predicting interpolation of features with a categorical feature, Softmax Regression is used.

- Linear regression: Since there is no real feature y to predict here, we can try to fit a model of the form $b + w_1x_1 + w_2x_2 + w_3x_3 = 0$. This captures a linear relation among the x itself.

Using the weights and bias values, all the features can be predicted if rest 2 features are known as follows:

$$\begin{aligned} x_1 &= (-b - w_2x_2 - w_3x_3)/w_1 \\ x_2 &= (-b - w_1x_1 - w_3x_3)/w_2 \\ x_3 &= (-b - w_1x_1 - w_2x_2)/w_3 \end{aligned} \tag{1}$$

The model is fitted using stochastic gradient descent for updating weight and bias. The model optimizes mean square error.

$$err(f) = \frac{1}{m}\sum_{i=1}^{m}\left(f(\underline{x}^i) - y^i\right)^2 \tag{2}$$

- Softmax Regression: Softmax regression is used for multi class classification of categorical variables. Instead of binary classes in logistic regression, softmax regression computes probability of a data point belonging to multiple classes, say C. The output is a one-hot encoded vector $y = (0, 0, \dots, 1)$. The cross entropy loss is minimised here -

$$\begin{aligned} err(f) &= \frac{1}{m}\sum_{i=1}^{m}\left[-\sum_{c=1}^{C} y_c^i ln[F_c(\underline{x}^i)]\right] \\ F_c &= \frac{e^{\underline{w}_c\cdot\underline{x}}}{\sum_{c'=1}^{C} e^{\underline{w}_c\cdot\underline{x}}} \end{aligned} \tag{3}$$

where F_c are non - negative and sum to 1.

The method followed for generating missing data ensures that on an average, $\sim$ 3K data points are missing for each feature. It is unlikely to have full data for any of the features. As a starting point we start filling up features with least amount of missing data. The steps followed are -

- Identify the feature with least percent of missing data out of the features which are yet to be filled, say x_1 and next feature with least percent of missing data, say x_2
- Filter out the rows with values missing for the selected feature, x_1. Use the remaining data points for training of x_2 given x_1 as input, using the algorithm of linear regression if x_2 is numerical or softmax regression if x_2 is categorical.
- Use the model results to fill in value of x_2.
- Identify next feature with least percent of missing data, say x_3. Use x_1 and x_2 to fill up x_3.
- Repeat these steps to fill up all of the features.

2.1.4 Linear Regression

In the methods described above, we are predicting one missing feature at a time. We now move on to a model where the input space is the entire data matrix with missing features and the output space is the data matrix with real data. We represent the missing values in the input with zeros and use the isMissing encoding while predicting the output. We represent the categorical features using one-hot encoding where each class will have a weight value associated it to.

As compared to the previous model, the relationships between all the features is learned at once in the form of a weight matrix. Thus, the input data matrix has dimensions (m, k), the true data is used as the output matrix for learning which has dimensions (m,k). We augment the bias value as 1 in the data feature. Thus, the weight matrix learned has dimension (k+1,k+1). The ideal weights are determined using the following matrix transformation:

$$W^* = [X^TX]^{-1}X^TY$$

Using the weight matrix, we predict the missing feature using the linear equation as described in Eq:1. The model is evaluated by calculating the root mean square error between the predicted feature and the true value of the feature. This model is much faster in terms of computation and all the linear dependencies are learned at once instead of creating k-regression models.

To avoid the weights becoming very large and prevent overfitting, we implemented ridge regression. The loss function has an additional norm of weight term associated to it with the hyperparamter λ. The optimal weights are then calculated using:

$$W^* = [X^T X + \lambda I]^{-1} X^T Y$$

However, the testing error is least for $\lambda = 0$ suggesting that the weights did not get too large in the normal regression. We can see the effect of ridge regression with varying λ in Figure 5.

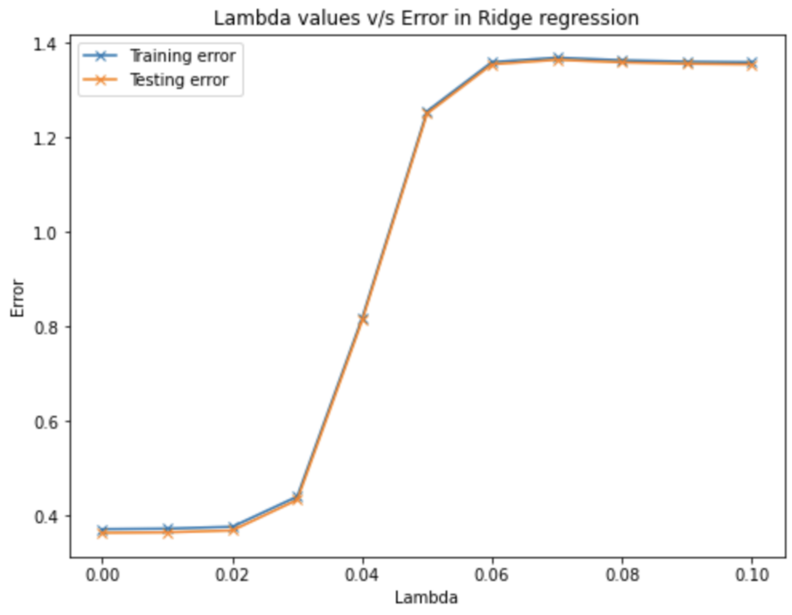


Figure 5: Training and testing error for different λ in ridge regression.

2.1.5 Neural networks

We extend the idea of reconstructing the data point by predicting the missing feature using a model where the entire data X is given as input and the target Y is the true data point. Earlier we considered linear regression and logistic regression models. These models are able to capture linear relationships between the features. However, as we saw in the data analysis, not all features are linearly correlated. This gives us the motivation to move to the non-linear space.

We implement a neural network where similar to the above method, we give the entire data point, with missing values as zero as the input. We map the input to the true data point forming an encoder network reconstructing the true data point. The reconstructing mean square loss is used as loss function and a stochastic gradient descent to learn the weights. The hyperparameters and features of the model that give the best testing and validation error are given in 2

The activation function of ReLU, sigmoid and tanh are employed along with weight initialisation techniques of Standard normal, Xavier and He. The results are shown in Figure 6.

Hyperparameter	**Value**
Learning Rate (α)	0.01
Activation Function	ReLU
Weight Initialization	Standard Normal
Number of layers	2
Number of hidden nodes	10
Batch Size	1 (SGD)

Table 2: Count of categorical feature classes and count of categorical features after pruning classes with less data representation

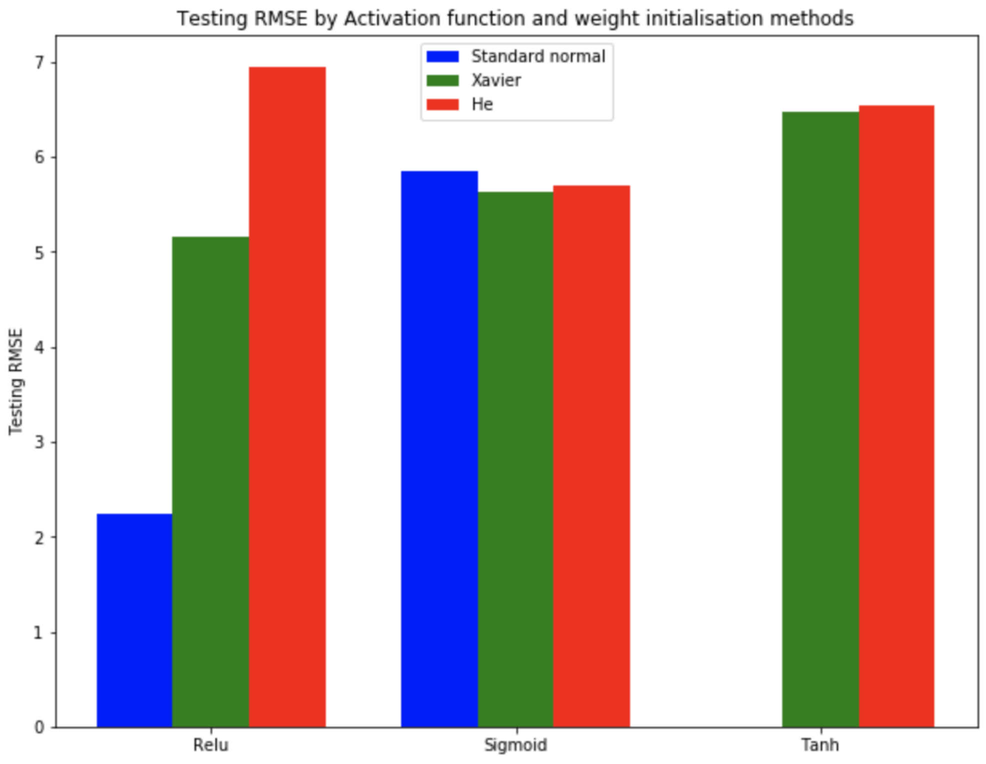


Figure 6: Testing RMSE for activation function and weight initialisation techniques.

2.2 Model Validation

Model validation is important in avoiding overfitting in the data and to determine the optimal hyperparameters. In our linear regression model, we use ridge regression to check if weights are in appropriate range. The analysis is explained in section 2.1.4

In neural network model, we divide the data into train set, validation set and the test set. We tune the hyperparameters of the model using validation set in a grid-search manner. We fit the data on the train set and evaluate its performance on the validation set. Once the hyperparamters are set, we test the model's final performance on the unseen test.

We also use the validation set for early termination. We calculate both the training and the validation loss based on the mean square error between the predicted data and true data. During training, if the validation loss does not decrease after n-epochs, we stop the learning even if the training loss keeps decreasing. This is to avoid overfitting. The loss function for both training and validation data can be seen in the following plot. We observe that after certain epochs, the validation loss starts to increasing, indicating overfitting.

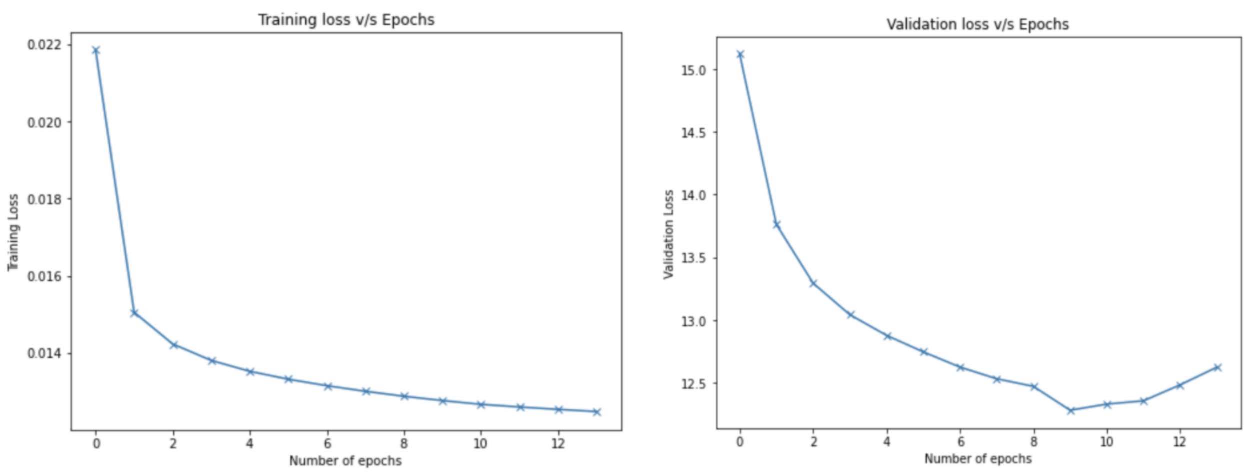


Figure 7: Training and validation loss in neural network as number of epochs increases.

2.3 Model Evaluation

The models implemented for interpolating/predicting of features are evaluated using the metric of root mean squared error. Since the categorical attributes are one-hot encoded, RMSE error is a valid metric for both numerical as well as categorical attributes.

$$err(f) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(f(\underline{x}^i) - y^i\right)^2} \tag{4}$$

Table 3 shows the testing RMSE for the models implemented. The linear regression model where in the input space is the (m x k) matrix of missing features and the output space is (m x k) matrix of real data gives the least testing error. Compared to the basic completion agent, the RMSE is decresed by 62.84% for the testing data.

Model	**Testing RMSE**
Baseline	0.977
KNN (n = 100)	0.980
Sequential Prediction	1.155
Linear Regression	0.3109
Ridge Regression ($\lambda = 0.01$)	0.432
Neural Network	2.245

Table 3: Performance of models for interpolating/predicting missing features when 10% of data is missing.

In order to analyse the amount of features needed for the model to interpolate well, the data is modified such that 5% and 20% data is missing and the models are run on it. The result is shown in Table 4. As per intuition, the increase in missing features leads to a increase in error of each of the models.

Data points missing in training	**% data missing**	**Testing RMSE**		
		Baseline	**Sequential Prediction**	**Linear Regression**
212923	6.67	0.803	0.949	0.207
322027	10.07	0.977	1.155	0.3109
644464	20.15	1.381	1.632	1.943

Table 4: Comparison of performance of models with increase in % missing data.

On taking the individual error of the columns, it was found that the RMSE of the column $Is\_income\_ < 50K$ is very high in comparison to prediction error of other features. The RMSE of income is 38, whereas

RMSE of other features is close to 0. For almost all the instances the missing income column is predicted as $Is\_income\_ \geq 50K = 1$. One reason for this could be the bias present in the data. Two third of the data is for white males residing in the United States. Due to this the model might not be able to generalise well to diverse data points.

3 Generate Data

We generate new data using explored models. The approaches to generating new data is as follows:

- Using feature sample parameters - Similar to prediction with baseline model, we use the sample parameters such as mean, variance and mode to generate new data. For each numerical feature, we estimate the mean and variance from the data.We use these parameters to generate new data from a normal distribution with estimated mean and variance. For categorical features, we estimate class probabilities for each category. We use this class probabilities to generate new categorical data.
- Using linear combination of features - In this method, we use the weights derived from linear regression to introduce linear relationship between the features. Each individual feature is generated using distribution with sample parameters (mean and variance for numerical, class probabilities for categorical). The features are multiplied by the corresponding weights such that they follow the linear model.
- K-means clustering - Another approach was unsupervised model of K-means clustering. Similar to the Euclidean distance metric and notion of similarity between data points using in K-Nearest Neighbor Regression, we group data into clusters based on the similarity between features. Once, the k clusters are derived, we have k - cluster centroids. Based on the mean, variance and class probabilities of cluster centroids, we can simulate more data for each cluster.

To compare the generated data from the models to the actual data, we compare the distributions. The sample parameters of the data can be compared easily. The mean, variance and class probabilities of the original data and the data generated using model1 and model 2 are almost same. Also, the likelihood of data can be determined for each model and compared.