S1 Principles of Data Science Coursework Report

CRSiD: tmb76

University of Cambridge

Section A

Question 1

The aim of this question is to explore the dataset and conduct some processing of the data. Conducting a check for missing values, which there were none, the data is explore in the following ways.

(a)

The data first needs to be visualised in some way. Now because there are 500 features, and therefore 500 dimensions in the feature-space, this means that for now, only visualisations of each feature separately is useful. Thus a density plot of the first 20 features is obtained and shown below (Figure 1). The main observation is that some features are very similar to each other, following similar distributions. A rough assumption is there are overall 34 groups of highly correlated features for the entire dataset. The most common one has a very strong peak at 0, with a much smaller one at around 1. Another is bimodal, with peaks at 0 and 3. And the last one is also bimodal, with peaks at 0 and a stonger one at approximately 4. If the features can be grouped then one could reduce the dimensionality of the feature-space to 1 representative feature from each group.

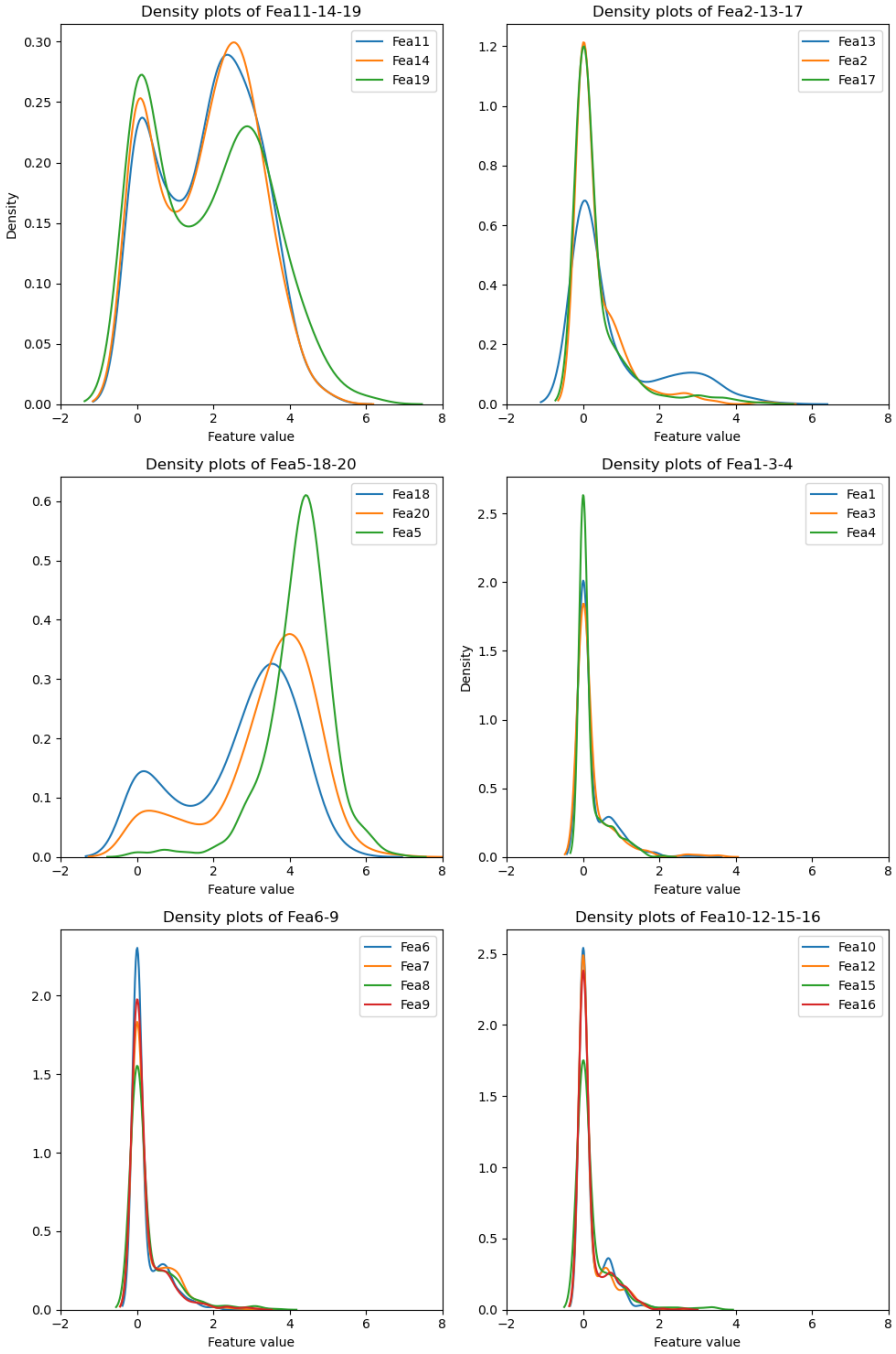


Figure 1: **Density plots of the first 20 features, grouped by similarity**. The x-axis was set to the same scale for all plots, for easire comparison. The y-axis is however different for all plots.

(b)

Thus, Principle Component Analysis (PCA) is conducted on the dataset. PCA enables one to derive a lower-dimensional set of features from the full **A** dataset. This is done by finding the direction in which the observations have the greatest variance, for the full feature space [1, pp. 255-257]. This is done on the entire dataset, not just the first 20 features. Using `scikit-learn`'s PCA function, getting the first 2 Principles Components (PC). The dimensionaly reduced dataset was obtained and the data was plotted in the following scatter plot (Figure 2).

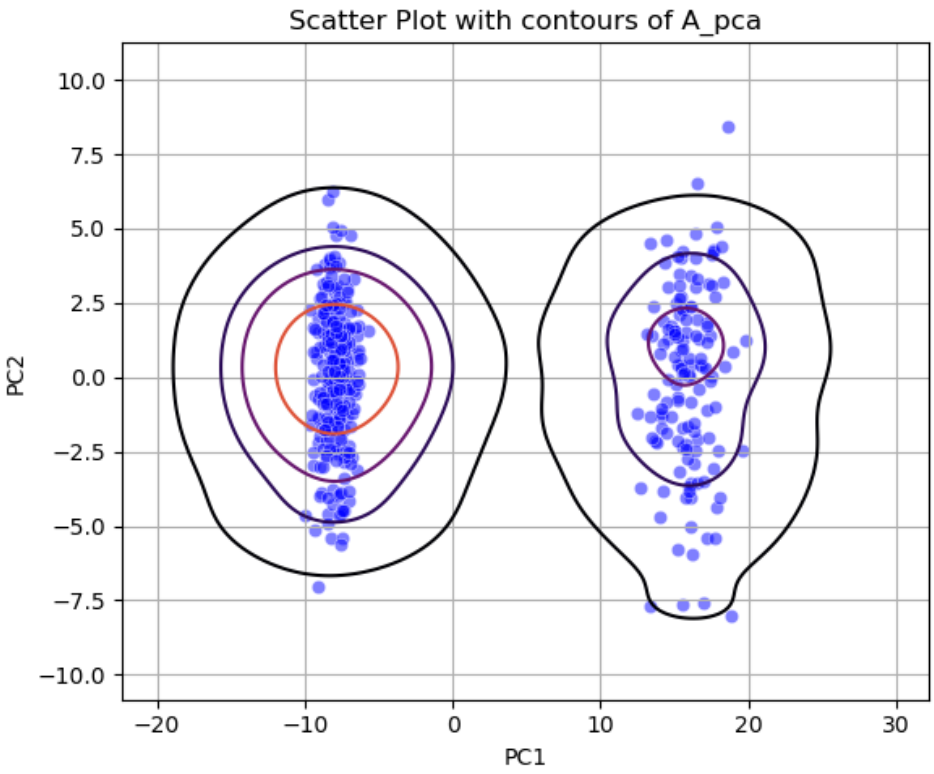


Figure 2: **Scatter plot of the dataset after PCA.** Density contours are also shown.

From Figure 2, the bimodal separation seen in some of the first 20 features is seen again in the 1^{st} PC. SMTHG about linear sep. The 2^{nd} PC also resembles the more commonly observed feature in Figure 1, with a very strong peak and a much weaker one at a slightly higher value, more from the right-hand side group. This is hard to see in the scatter plot, but the density contours do show the negative skewness of the data's distribution for that PC.

(c)

From this, clustering can be run to try and see how they would compare to the distribution observed. The `scikit-learn` KMeans clustering algorithm was used, with all parameters set to their default values, apart from the random state. The default number of clusters is 8. Now from Figure 1 and figure 2, it seems unlikely that there are 8 clusters in the data. This is one of the risks of K-means clustering as it will "find" 8 clusters, even if there are not 8 clusters in the data. To assess if "default" K-means clustering performs adequatly, the data is split into 2. For both these splits, clustering is run and then applied to the other split. The reason this can be done is that K-means clustering's main result is the position of points called centroids, which are the geometrical centres of each clusters (when defining the centroids from the feature means). In terms of their significance they are each the most representative points of each cluster. One of the `scikit-learn` K-means output is those `cluster_centers_`. Then when using the `.predict` method after fitting the model, i.e. clustering one half of the date, the other half can be clustered base on the distance of each point to the centroids of the clusters. This is done for both splits of the data and the results are shown in the tables below:

Clusters	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	67	0	0	0	0	0	0
4	0	70	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0
6	1	0	11	2	9	11	26	3
7	0	0	1	1	1	0	0	0

Table 1: Contingency table for the 2 clusterings on the first split of the data.

Clusters	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	76	0	0	0	59	0	0	0
2	0	0	0	0	0	0	1	2
4	0	1	0	0	0	1	0	0
5	0	0	1	0	0	3	3	1
6	0	1	0	2	0	11	34	8

Table 2: Contingency table for the 2 clusterings on the second split of the data.

(d)

As can be seen, cluster 6 for the 2^{nd} clustering (columns) is empty. This can be explained by the initialization of the centroids. Indeed the K-means clustering has

to place the centroids somewhere at the start, and they are then updated based on which points are closest to it and their mean. If the centroid for cluster 6 in the 2^{nd} clustering happened to have no points closer to it than other centroids, it stayed where it was for the whole clustering process [**?**]. Overall, the fact that a too large cluster number was selected for the model means there is almost no agreement between clusters (Table 1). This means that the clusters are not very stable. Stability here refers to getting similar clusters when clustering the same or very similar datasets. Furthermore, the number used to designate the cluster is not the same in the 2 clusterings. In other words, it could be the case that the relatively greater agreement between cluster 1 from the 1^{st} clustering (rows) and cluster 2 from the 2^{nd} clustering (columns), is due to the fact they have centroids that are close to each other. But they happened to have different numbers. Regardless the agreement is small since all values in the table are small.

Thus, we repeat the clustering for a lower number of clusters, 2, based on the visualisation of the data, Fig 2, and ignboring the actual knowledge of there being 3 clusters. We get the following results:

(e)

Here, with the 2 clusters, both models agree on how to separate the data.

Conducting PCA before and after clustering is useful in both cases. One gives a clue as to how many clusters one should use in the K-means training and the other enables one to visualise the clustering results. One additional thing is that the K-means could have been done on the PCA reduced dataset, which would have been a 2D dataset. The impact on performance in that case is not completely clear.

Question 2

The same dataset as in A is given. But here, the aim is to deal with duplicated observations and missing labels there are in this version of the dataset.

(a)

First, the frequency of the labels is summarised in the table below (Table 2):

(b)

Taking only the feature columns, the following rows were identified as duplicates:

One way to address the dupicates is to use a supervised learning method to train a model on the non-duplicated rows and then predict the labels for the duplicated

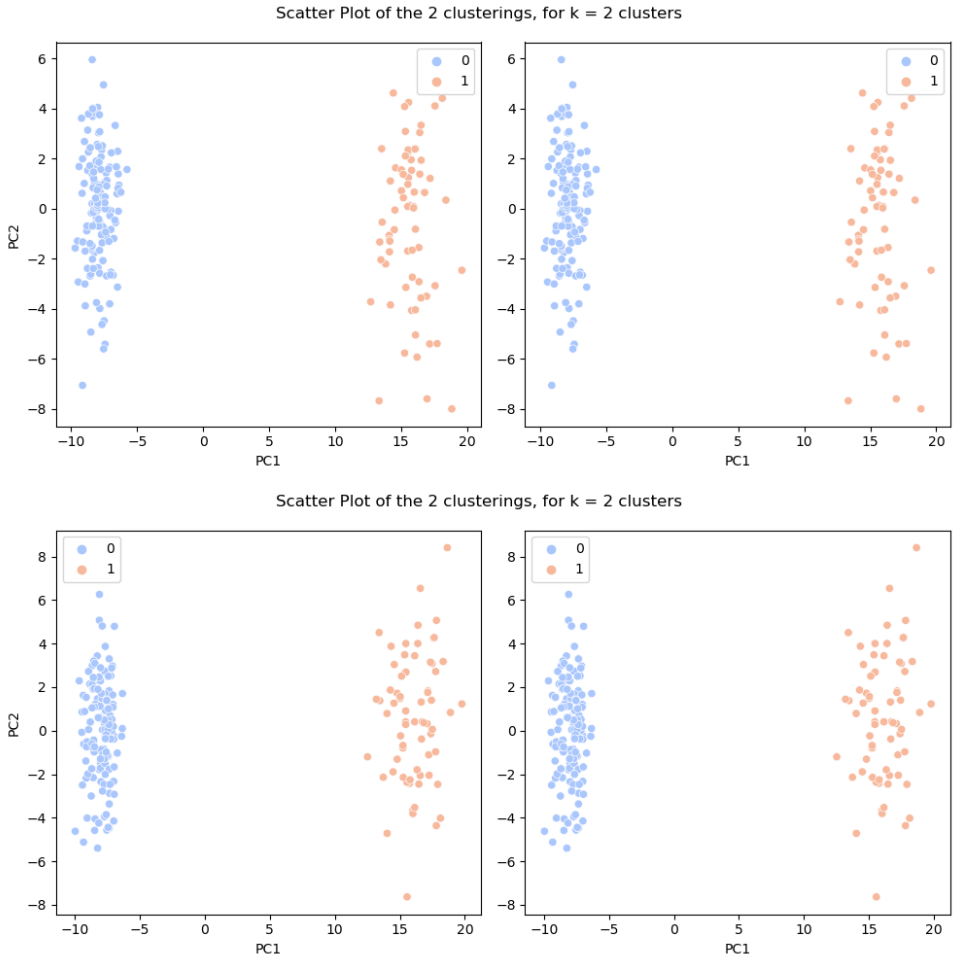


Figure 3: **Scatter plot of the 2 clusterings** The colors indicate the cluster membership.

rows. Once we obtain these label predictions, we compare them with the duplicates' labels and see which ones match. If one does, then we keep that row and discard the other. If both match, we drop one of them to avoid having duplicates, even correctly labelled ones. And if none match, we drop both rows.

Using this method, the following rowws were dropped:

Leaving 16 duplicated rows but correctly labelled. In this case we dropped half of those, dropping samples: [106, 145, 192, 248, 252, 310, 388, 423].

Label	Frequency
1.0	179
2.0	157
4.0	72

Table 3: Frequency of the labels in the **B** dataset.

Samples: [48, 50, 64, 66, 86, 94, 103, 120, 121, 127, 137, 139, 140, 166, 167, 186, 193, 195, 208, 213, 218, 230, 240, 269, 273, 280, 311, 317, 325, 331, 364, 371, 372, 379, 402, 404, 409, 416, 429, 444]

Table 4: List of duplicated rows.

(c)

Missing labels is sometimes referred to as a semi-supervised learning case [1, p. 24].Here are 2 ways observations with missing labels can be dealt with. The first is model based imputation. This relies on training a machine learning model on the dataset with observations with missing labels removed, and then predicting the labels for these observations. The choice of model here is broad since the total number of missing values is not too great, and multiple models would be appropriate. This method takes into account relationships between features though runs the risk of an invalid prediction and therefore affecting the result of any model training conducted subsequently. Another more brute-force approach is a simple removal of those observations. This has the advantage of simplicity though brings loss of information, and becomes inappropriate in a scenario where many observations have missing labels. For that second method especially, it is important to consider why the labels are missing. They can be Missing At Random (MAR), meaning there is no underlying reason for that value missing. On the other hand they could be Missing Not At Random (MNAR), meaning that the data missing, sometimes combined with the values of the features for that observation, carry an important meaning. In that case, dropping these observations would be an important loss of information [1, pp. 515-516].

(d)

Here, a model based imputation method is used, assuming the labels are MAR. A multinomial logistic regression is trained on the data without the rows with missing labels. Those 20 rows are dropped, the model is trained and testing returns an accuracy of 0.847. The model is then used to predict the labels for the rows missing them. We obtain new frequencies for the labels:

Samples: [29, 82, 100, 116, 118, 146, 172, 209, 219, 259, 290, 296, 350, 358, 381, 395]

Table 5: List of duplicated rows.

Label	Frequency
1.0	187
2.0	162
4.0	64

Table 6: Frequency of the labels in the **B** dataset, after missing label prediction.

Table .. shows a slight tendency to predict labels 1.0 and 2.0 compared to 4.0, though there was no strong proof the observations with missing vlaues would follow the original proportions.

Question 3

a

In this dataset, values in various observations and features are missing. 5 observations have missing values. These are samples: [137, 142, 230, 262, 388]. For each of these rows, the following features are missing:

Missing Features: [Fea58, Fea142, Fea150, Fea233, Fea269, Fea299, Fea339, Fea355, Fea458, Fea466, Fea491]

(b)

Again, one can deal with these missing values in a number of ways. One is to use model-based imputation, as in Q2. An example of that is using a k-Nearest Neighbour imputing method. The model looks at the k closest observations to the one with missing values and takes the mean of the values for that feature. Again, this captures relationships between the features, and one can select the number k as they wish. However, it does assume a linear relationship. Another is a simple constant value-based imputation. This usually just takes the mean or median value of that feature and imputs it where value are missing for that feature. This is a simple approach where you can avoid dropping values but it ignores any relationship between features and as a result some loss of information occurs. One way to improve this result is by conducting mulitple imputations. Choosing an imputation model, one can repeat the imputation, obtaining a mean and uncertainty of their results. Moreover, imputing multiple times means greater adaptability to the data [3].

(c)

Here, a k-Nearest Neighbour imputation is used. As said above, this takes into account feature relationships while not being too computationally expensive. The number of neighbours k is set to 10. This is from training a kNN model in Q2 and found an optimal number of neighbours to be 10. Comparing the distribution within the concerned features before and after imputation is the plot below:

0.0.1 (d)

To detect outliers, one can calculate the z-scores of each features by standardizing the data. This is done by shifitng/subtracting feature values by their mean and dividing by their standard deviation [2, p. 73]. This thus expresses each value by how many standard deviations they are from the mean. By then setting a threshold of 3, values furhter than 3 standard deviations can be identified.

0.0.2 (e)

Multiple K-NN imputation.

Question 4

(a)

A single decision tree is grown through recursive binary splitting of the data. It is a greedy search as at every node, the splitting is only made according to what the best split is at that time [1, pp. 337-338]. That is described by some criterion, like the Gini Index. The Gini index is a measure of the node "purity", how much of one class the node contains [1, pp. 338-339]. Once the tree is fully grown down to having only totally pure nodes at the end, one can prune it, removing some of the last nodes to avoid having an overtrained tree. In bagging, multiple trees are grown independently on bootstrapped samples of the dataset and combined to reduce the high variance issues single decision trees have [1, p. 343]. Random forests are built in a similar way to bagging but as a tree is built on a bootstrapped sample, at each split the tree has to make, it is only performed on a subset of the features. This decorrelates the trees, and let's them explore a lot more possible splits [1, 354]. This sort of helps counteract the greedy nature of the tree. When using a random forest there are a few hyperparameters one can tune to control how each tree is grown. First is the criterion used, as there are options other than the Gini index. Second, the number of features that the tree has to split from at each node. It is usually set to the squareroot of the total number of features.

(b)

In preparation to training a classifier on the data, some preprocessing was done. The first thing done was to count missing values, and none were found. Then, checks were run for duplicates and none were found either. Now, checks were made for zero-variance features and the following features were identified and dropped:

0 Variance features: [Fea49, Fea66, Fea94, Fea97, Fea106, Fea109, Fea125, Fea129, Fea151, Fea152, Fea187, Fea189, Fea223, Fea224, Fea238, Fea264, Fea293, Fea324, Fea384, Fea432, Fea440, Fea450, Fea463, Fea543, Fea636, Fea666, Fea680, Fea701, Fea707, Fea728, Fea729, Fea750, Fea785, Fea787, Fea808, Fea830, Fea846, Fea930, Fea943, Fea945, Fea973, Fea982]

Further, with the average variance being 0.618, 16 rows were found with near-zero variance and were also dropped:

Near-zero Variance features: [Fea747, Fea730, Fea57, Fea544, Fea4, Fea154, Fea597, Fea610, Fea19, Fea855, Fea347, Fea843, Fea499, Fea231, Fea404, Fea546]

Outliers were then dealt with

Finally, 2 pairs of highly correlated features (¿ 0.9) were found and one of each pairs were dropped: Feature 300 and 345, and Feature 869 and 954.

0.0.3 c

The random forest classifier was trained on default parameters: 100 trees, Gini index splitting criteria, no max depth set, and the max number of features to split from set to the squareroot of the total number of features.

Training this model, a test set classification error of 0.05 is found.

(d) Optimise the algorithm with respect to the number of trees in the random forest. You should be able to do this without explicitly performing cross-validation.

(d)

This number of trees in the random forest can be optimised using an Out-Of-Bag (OOB) score. This works from the fact each tree is built for a bootstrapped subsample of the dataset. Leaving an "Out-Of-Bag" sample the trees for which that sample was OOB can be tested upon. An OOB classification error can thus be obtained, for each number of trees [1, p. 345].

Thus, the optimal number of trees is around 170-200, where the OOB error stabilizes, and going further only makes the model more computationally expensive.

Re-running the model for the optimal tree number, the test-classification error now is 0.04.

(e) Calculate the feature importance. In your report, describe and interpret the feature importances. Retrain the model using a subset of the most important features. In your report, indicate which features you have chosen and compare the retrained classifier with the original classifier.

(e)

One characteristic of the random forest classifier is it can give us the relative importance of features. It is computed as how much of a reduction in the Gini index (or other used criterion) is brought by that feature. Were it not for the bootstrapping and subset of features to split from, there would not be enough information for that measure to be computed accurately [1, pp. 345-346]. In this case, we obtained the following plot, with the 50 most important features:

We see that feature importance decreases very rapidly before steadily decreasing slowly. Taking the first 20, another random forest model can be trained. With this model, a test set classification error of 0.0999 was obtained. As expected, the performance is slightly lower but considering the amount by which computational complexity was decreased, this is a very positive result.

(f) Repeatsteps(b),(c)and(e)foroneothersupervisedlearningclassifier. Compare and contrast the two classification approaches and your results for each.

Question 5

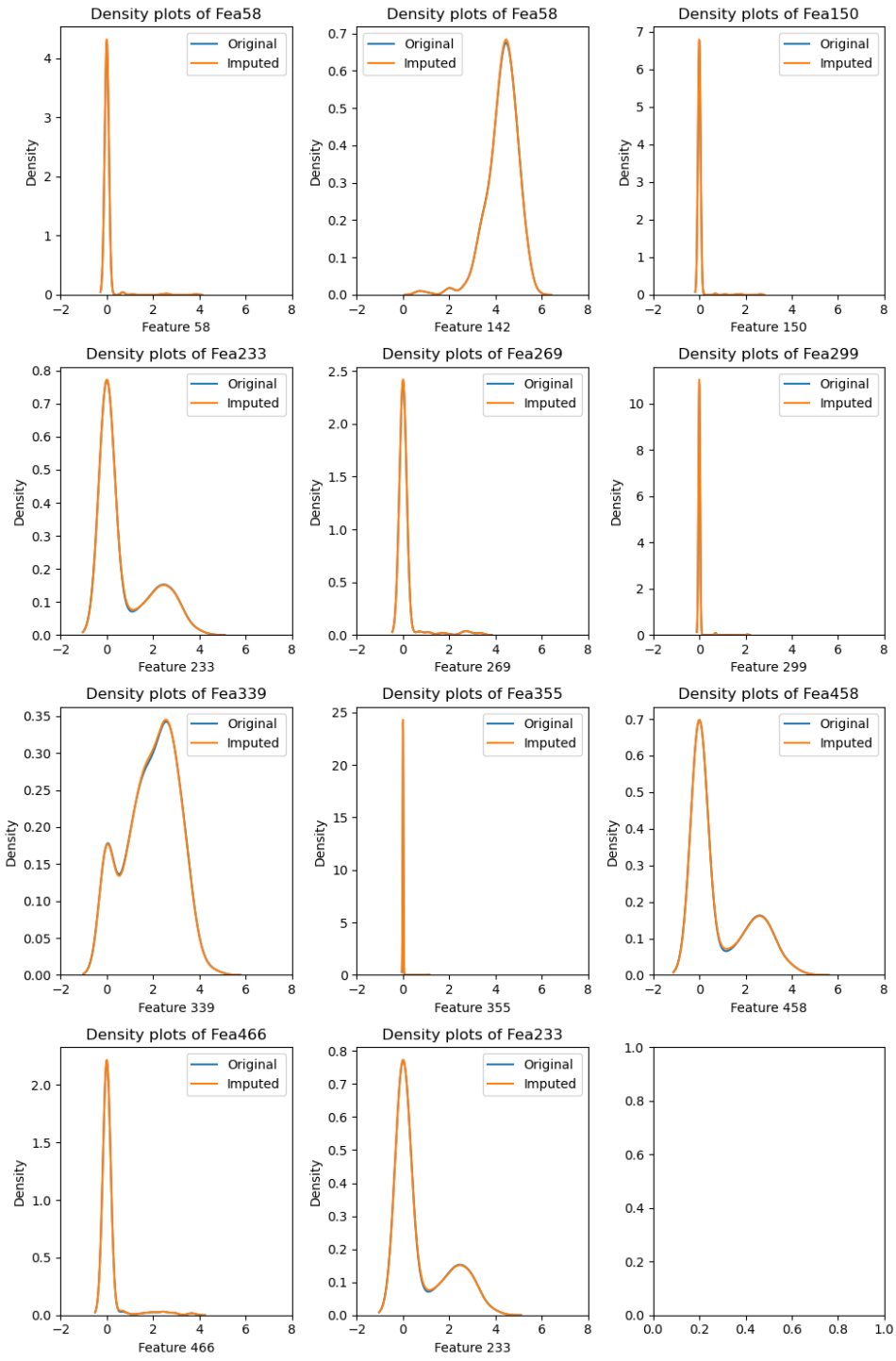


Figure 4: **Density plots of features before and a** Density contours are also shown.

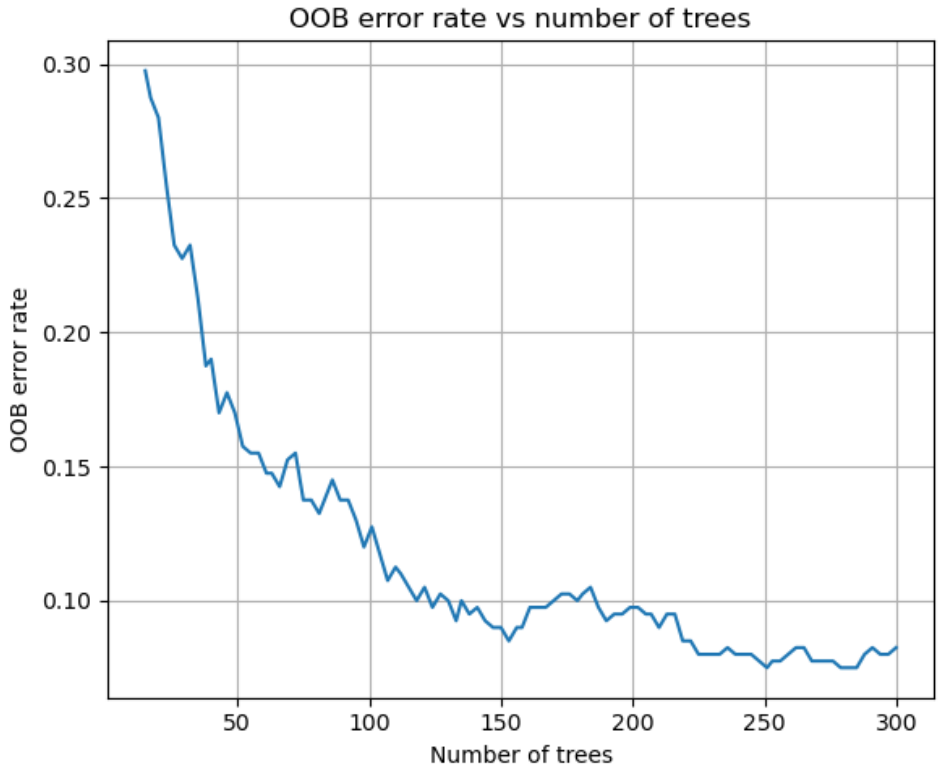
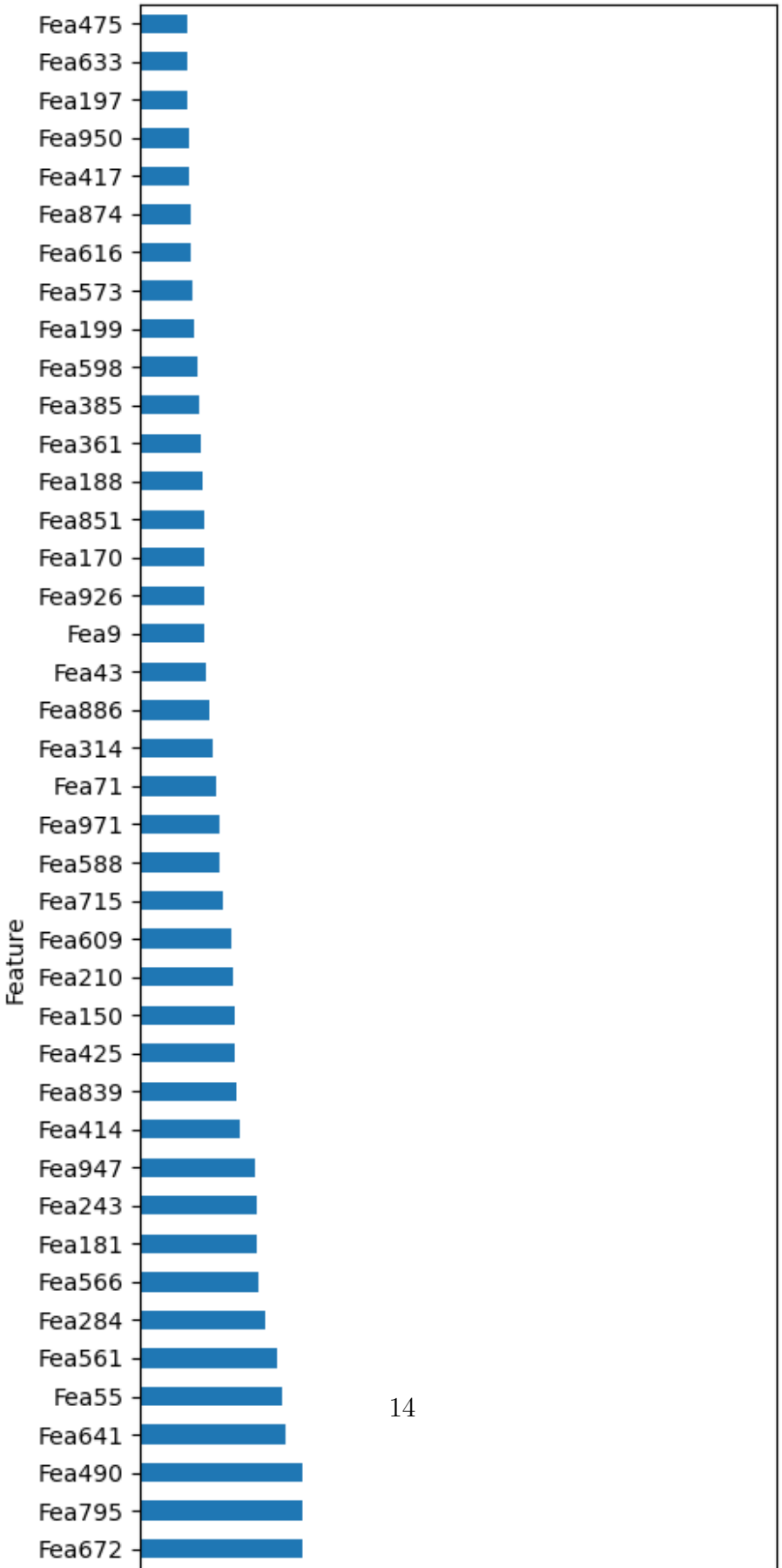


Figure 5: **OOB error vs number of trees** Density contours are also shown.


Feature
Fea475
Fea633
Fea197
Fea950
Fea417
Fea874
Fea616
Fea573
Fea199
Fea598
Fea385
Fea361
Fea188
Fea851
Fea170
Fea926
Fea9
Fea43
Fea886
Fea314
Fea71
Fea971
Fea588
Fea715
Fea609
Fea210
Fea150
Fea425
Fea839
Fea414
Fea947
Fea243
Fea181
Fea566
Fea284
Fea561
Fea55
Fea641
Fea490
Fea795
Fea672

Bibliography

[1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning with Applications in Python*. Springer Texts in Statistics. Springer, 2013.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. scikit-learn: Machine learning in python. `https://scikit-learn.org`, 2011.

[3] Jerome P. Reiter. Statistical analysis of network data with r. `http://www2.stat.duke.edu/~jerry/ReiterUNC2020.pdf`, 2020.