Cem Daloğlu
CS464-1 HW2

1) PCA on Van Gogh Paintings

Question 1.1

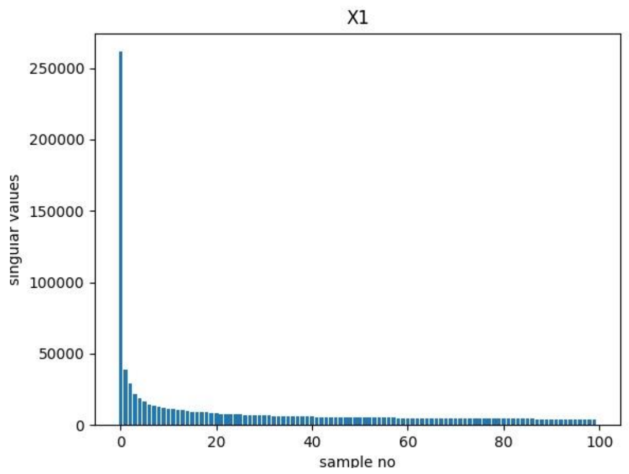

Figure 1: Bar chart of first 100 singular values for X1

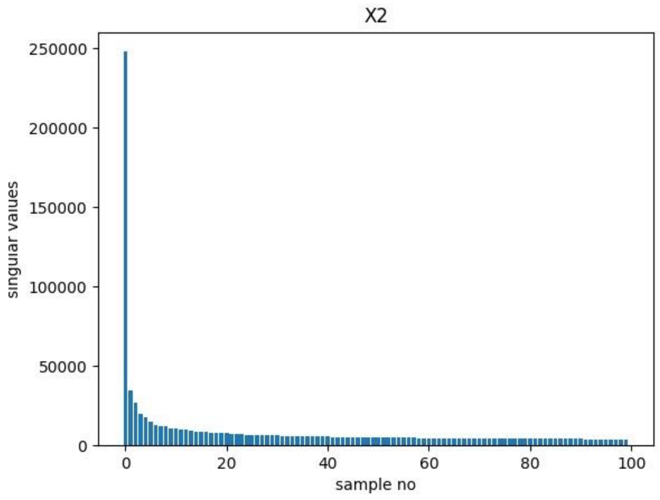

Figure 2: Bar chart of first 100 singular values for X2

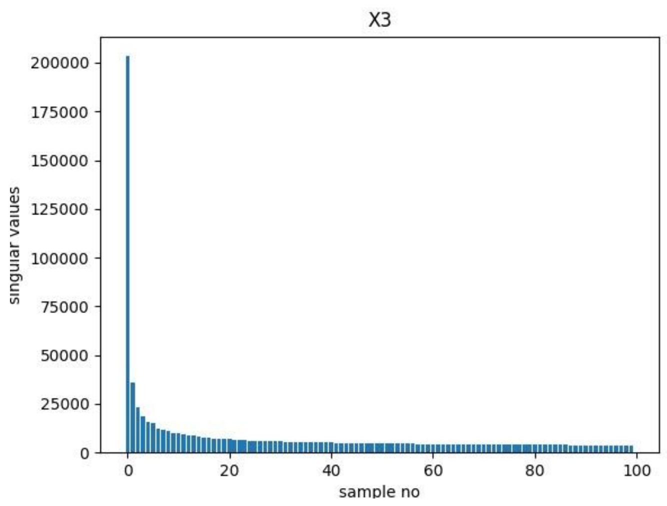

Figure 3: Bar chart of first 100 singular values for X3

By looking at the above figures, it could be said that first 20 values can be named as principal components since they are higher than the rest.

Proportion of Variance Explained (for first 10 principal components):

$\boldsymbol{X_1}$: [0.85965542 0.01884455 0.01075795 0.00610412 0.00433712 0.00353375

0.00247041 0.00235895 0.00212203 0.00173626]

$\boldsymbol{X_2}$: [0.86155932 0.01668933 0.00989828 0.00553156 0.00430903 0.00322918

0.00227687 0.0021419 0.00201153 0.00164281]

$\boldsymbol{X_3}$: [0.82201994 0.02533863 0.01077714 0.00695336 0.00497029 0.00444568

0.00291419 0.00270662 0.00237997 0.00198714]

As it can be seen from the above PVE results for X_1 and X_2, the first principal component explains the %86 of variability. In other words, %86 of the variance retained in the first principal component. For X_3 %82 of the variability is explained by the first principal component.

Question 1.2

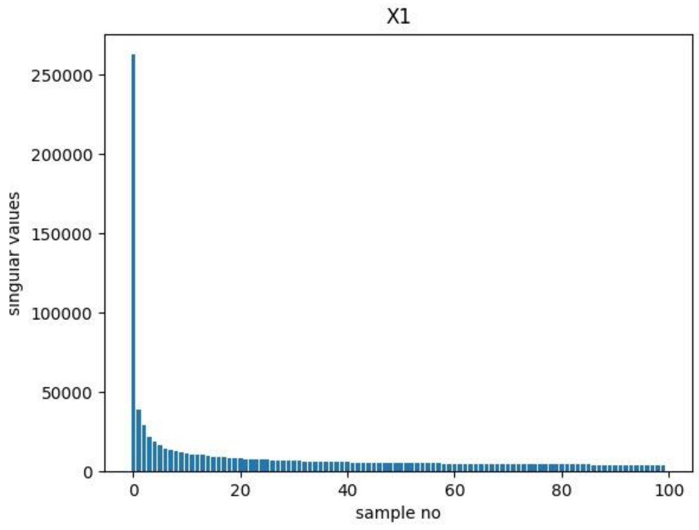

Figure 4: Bar chart of first 100 singular values for X1

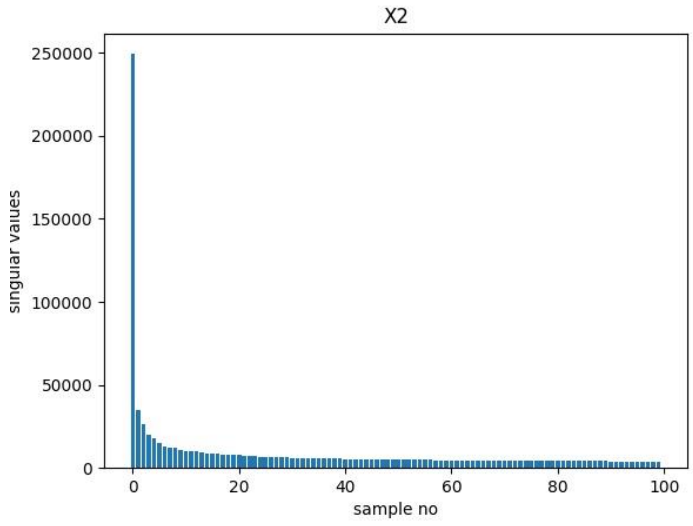

Figure 5: Bar chart of first 100 singular values for X2

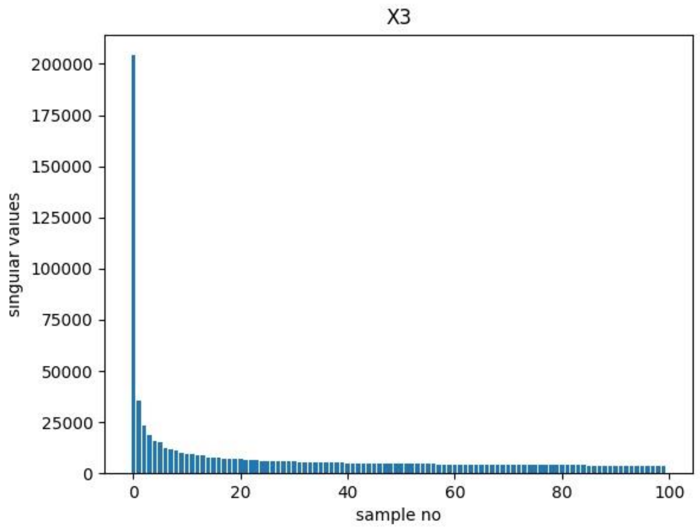

Figure 6: Bar chart of first 100 singular values for X3

By looking at the above figures, it could be said that first 20 or 30 values can be named as principal components since they are higher than the rest. Artificially added noise did not change the singular values significantly so it could be said that PCA eliminates or does not effected by noise (for the given parameters). Images can be reconstructed by multiplying the outputs of the SVD which are U, s and V^T. By deciding how many principal components to use for reconstruction, images can be reconstructed. When we use all principal components, I expect to see the same picture that is given as input. But if we use all principal components then, there is no point to apply PCA. Thus, 30 principal components is tried and following picture is obtained:

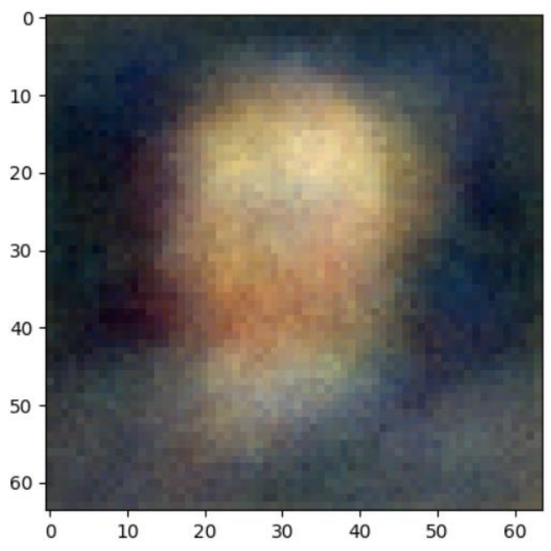

Figure 7: Reconstructed with 30 PC

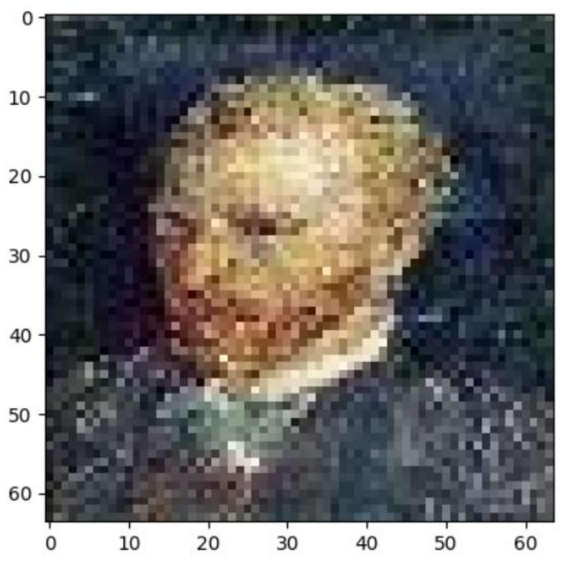

Figure 8: Original

When I applied 877 principal components PCA gave the below picture:

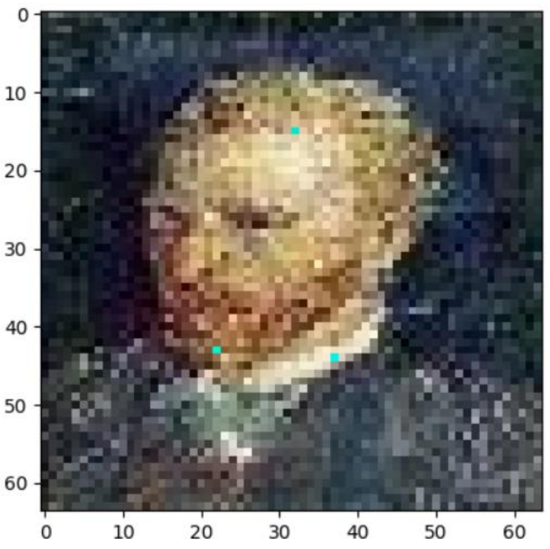

Figure 9: Reconstructed with 877 PC

Thus, as principal component number increases, reconstruction accuracy will increase, different numbers of principal components should be applied to obtain the best result with respect to image noise and principal component number. In other words, there is a trade-off between number of principal numbers and image quality. Small number of principal component should be selected in order to decrease the presence of artificially added noise. But, again there is a trade-off. When small number of principal components selected picture quality decreases as well.

2) Linear Regression on University Admission Records

Question 2.1

From the given Eqn. 2.1 after the given multiplication, the eq.1 is obtained;

$$J_n = y^T y - y^T X\beta - y^T X\beta + \beta^T X^T X\beta \quad (eq.1)$$

Taking derivative of eq.1 must be equal to zero and eq.2 is obtained in the following;

$$0 = \frac{dJ_n}{d\beta} = -y^T X - y^T X + 2X^T X\beta \quad (eq.2)$$

To get the β, eq.2 should be written according to β;

$$\beta = (X^T X)^{-1} X^T y \quad (eq.3)$$

X is feature matrix, y is output matrix and β is weight matrix.

Question 2.2

MSE:

[0.003573439267274022, 0.0031448956541765137, 0.004206339755471159, 0.003975786542517156, 0.003545311721155047]

MAPE:

[0.07242732297885611, 0.06389102623165059, 0.06006986161238634, 0.06421546367009678, 0.06225802086127264]

R^2:

[0.8472842142226144, 0.827453364829642, 0.834044772508528, 0.7179744858283528, 0.8354753363145404]

MAE:

[0.040278388243865544, 0.041076338515938435, 0.04440631589499839, 0.04571577385224034, 0.04491905287787124]

Question 2.3

MSE:

[0.004996513299254564, 0.004645126695579681, 0.004122843568026806, 0.00446551654608398, 0.003240231332636076]

MAPE:

[0.07485184385903655, 0.08401532955346352, 0.07603648668615864, 0.0942562535010897, 0.07625012795186399]

R^2:

[0.6452409856271211, 0.708807711357742, 0.7968113178630629, 0.6905332268528499, 0.7510097377586585]

MAE:

[0.04875698895749363, 0.0618437341566657, 0.0390677398922728, 0.05777301639020521, 0.05462491530155977]

Question 2.4

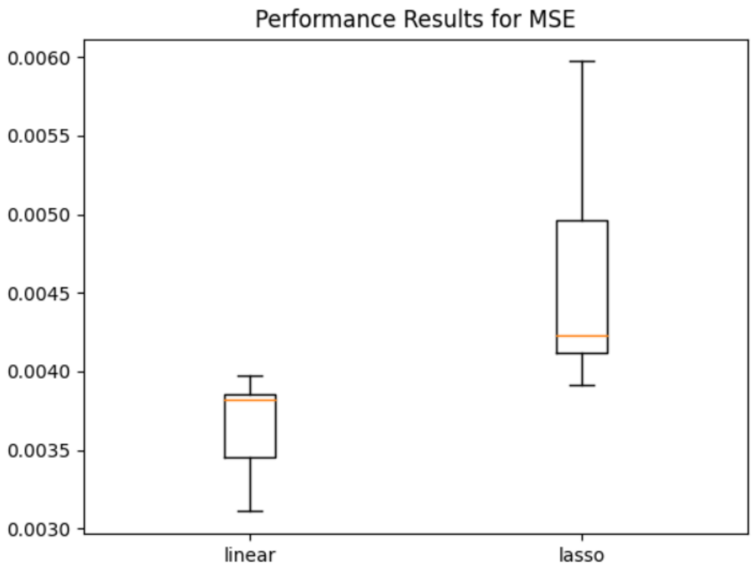

Figure 10: Box plot of MSE

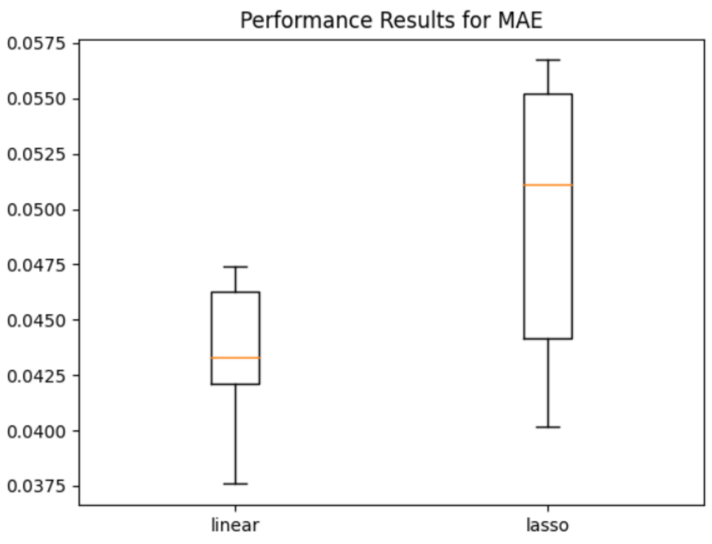

Figure 11: Box plot of MAE

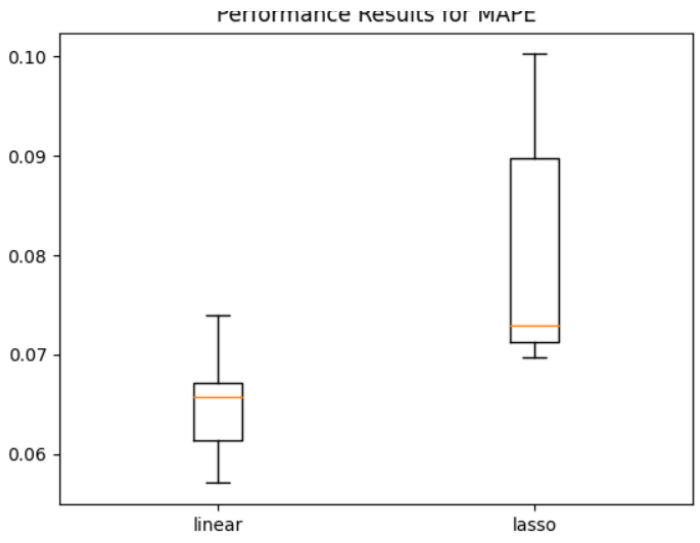

Figure 12: Box plot of MAPE

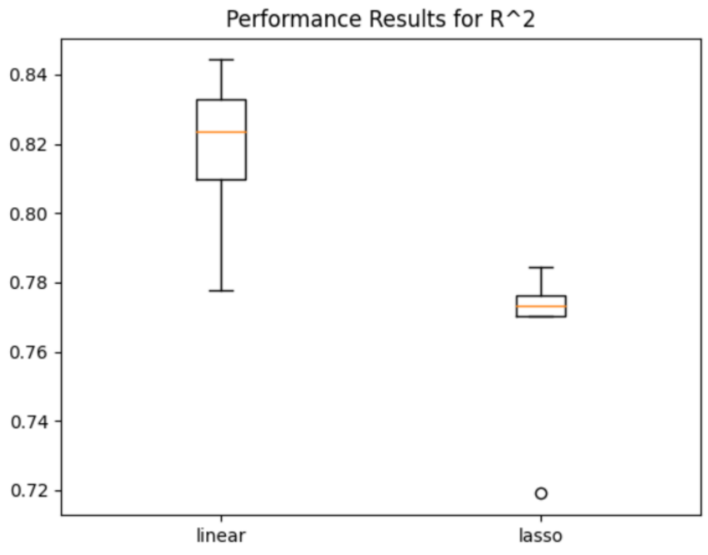

Figure 13: Box plot of R^2

From Figure 10, it can be said that linear model has less error than lasso model but there is no significant difference between these models. Same picture can be seen in Figure 11 as well but this time the difference between two models is higher than the MSE. In Figure 12, again lasso model is worse than linear model and also the difference between their errors is the highest among all errors. By looking at the Figure 13, R^2 of linear model is higher which means, predictions of linear model is closer to the regression than the lasso model.

Since MSE takes the square of the difference of the actual response and predicted response, it also increases the outliers' error. But MAE does not take the square of the error so MSE is more sensitive to outliers than MAE. From the Figure 10, two models performance seems similar to each other, there is a small difference between them. But in MAE (Figure 11), their difference is higher. Thus, by looking at Figure 11, it can be said that linear model is better than lasso model. Performance of the models are more similar compared in terms of MSE because the predictions are in between 0 and 1. So, when 0.2 difference occurs between actual and predicted value, MAE gives 0.2 error but MSE gives 0.04 error. In other words, because of the square in MSE, actual and predicted value difference becomes smaller than the difference.

I would choose MAPE performance metric since there is no 0 input for MAPE and it gives more accurate results than other metrics.

Since the given dataset size is small, we need to train our model as much as possible. Without cross validation, model would be trained for only 400 iterations but with the help of the cross validation, model trains $4 * 400 = 1600$ iterations which would give better results than not applying cross validation.

Since there is a correlation among features, L_1 regularization cannot make this correlation and ignores correlated features. Since in the given question, output depends on correlated features. When we apply lasso regularization, model would not be able to create correlations between features.

Cross validation would give better predictions at the end but it would take more time than fixed test dataset approach. Thus, in theory I would pick cross validation approach but in reality I would use fixed test dataset approach.

3) Logistic Regression for Survival Prediction

Features are normalized for both in the training and test set for better performance metrics.

Question 3.1

Learning rate is chosen as 0.01 since other values gave less accurate results (approximately %20 less accurate). Performance metrics are reported below.

Performance metrics:

accuracy: 0.8324022346368715

precision: 0.819672131147541

recall: 0.7246376811594203

NPV: 0.8389830508474576

FPR: 0.1

FDR: 0.18032786885245902

F1: 0.7692307692307692

F2: 0.7418397626112762

Confusion matrix:

Predicted vs. Actual	Survived	Not survived
Survived	50	11
Not survived	19	99

For 1000 iterations, every learning rate in the set is applied to the model. When the learning rate is 0.01, best performance is obtained. Model weights are converged when 1000 iterations applied since performance metrics do not change for shuffled data. It takes 0.8 seconds to train the model.

Question 3.2

Similar as the previous question, learning rate is chosen as 0.01 since other values gave less accurate results (approximately %20 less accurate). Model weights and performance metrics are reported below.

$$w_0 = Bias\ weight, w_1 = Ticket\ class, w_2 = Male, w_3 = Female, w_4 = Age$$

$$w_5 = Siblings/Spouse, w_6 = Parent/Children, w_7 = Fare, w_8 = Cherbourg$$

$$w_9 = Queenstown, w_{10} = Southampton$$

Model weights: $[w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}]$

100^{th}: [-0.07750196 -0.11223309 -0.16187306 0.08292962 -0.03874885 -0.00559845
-0.00163963 0.02680604 -0.07053232 0.01041259 -0.00948706]

200^{th}: [-0.10316484 -0.18944336 -0.27818704 0.17358072 -0.05763854 -0.00806352
0.0007702   0.03673659 -0.11060749 0.0288189 -0.01348107]

300^{th}: [-0.1040081 -0.2466797 -0.36776916 0.26231958 -0.06655351 -0.0092842
0.00401124 0.04740949 -0.12996908 0.04915572 -0.01529956]

400^{th}: [-0.09330836 -0.29336483 -0.44079556 0.34604573 -0.07079699 -0.01004654
0.00735702 0.05813679 -0.13898183 0.06963464 -0.016066]

500^{th}: [-0.07747051 -0.33412227 -0.50253061 0.42361862 -0.07293275 -0.01071873
0.01048091 0.06860892 -0.14270738 0.08945491 -0.01632286]

600^{th}: [-0.05962926 -0.37130066 -0.55590654 0.4948358 -0.07422061 -0.0114699
0.01324714 0.07869727 -0.14368934 0.10828906 -0.0163338]

700^{th}: [-0.04131455 -0.40612985 -0.60269232 0.55993629 -0.07528019 -0.01237209
0.01561254 0.08836078 -0.14323179 0.12603772 -0.0162253]

800^{th}: [-0.0232541 -0.43927341 -0.64404598 0.6193504 -0.07641055 -0.01344969
0.01757837 0.09760008 -0.14201242 0.14270898 -0.01605548]

900^{th}: [-0.00577208 -0.47110213 -0.68078657 0.67357301 -0.07774914 -0.01470405
0.01916614 0.10643463 -0.14038681 0.15835833 -0.01584842]

1000^{th}: [0.01100915 -0.50183256 -0.7135332   0.72310087 -0.07935212 -0.01612598
0.02040544 0.11489126 -0.13854302 0.17305883 -0.01561149]

Performance metrics:

accuracy: 0.8324022346368715

precision: 0.819672131147541

recall: 0.7246376811594203

NPV: 0.8389830508474576

FPR: 0.1

FDR: 0.18032786885245902

F1: 0.7692307692307692

F2: 0.7418397626112762

Confusion matrix:

Predicted vs. Actual	Survived	Not survived
Survived	50	11
Not survived	19	99

When the learning rate is 0.01, best performance is obtained. For the full batch gradient ascent algorithm, weights are converged when 1000 iterations applied. Convergence can be seen from the performance metrics of mini batch and full batch algorithms. Both algorithms performance metrics gave same results since both models reached their threshold iteration number for convergence. It takes 0.03 seconds to train the model.

Question 3.3

Highest positive and lowest negative weights gave the most important features for surviving and not surviving condition. Positive results contribute to surviving condition and negative values contribute to not surviving condition. Feature normalization is needed in order to create an objective feature importance. By normalizing, all feature values change between 0 and 1, if features do not have same boundaries, feature values with higher boundaries affect feature importance more than features values with lower values. Therefore, the range of all features should be normalized so that each feature contributes similarly to the feature importance or weights. It is possible to compare importance of categorical and continuous features together with the help of the one hot encoding. The two most important features are male and female features since in the weight values given above (3.2), $w_2(male)$ has the lowest negative value which means, if a passenger is a man, it is more likely him not to survive. $w_3(female)$ has the highest positive value which means, if a passenger is a woman, it is more likely her to survive. For the mini-batch gradient ascent algorithm, when I both increased and decreased the batch size, training time does not change significantly. Thus, batch size does not affect training time of the model.

4) SVM

In the following questions, 5-fold cross validation is applied and precision is selected as the performance metric for finding the best model. Since we do not know if class labels occurred same number of times, precision would give more objective results than accuracy. Thus, I select precision.

Question 4.1

Grid search is performed on the hyper-parameter space and according to validation set, hyper-parameter C is tuned and $C = 0.01$ was selected as the best model for this specific shuffled dataset. Since small C value (0.01) gave the best performance, it could be said that hard margin performs better than soft margin for this problem.

It takes approximately 70 seconds to run the setup for five different test set. Precision results' boxplot can be seen in the Figure 14 and precision results with selected C values for each test fold can be seen below.

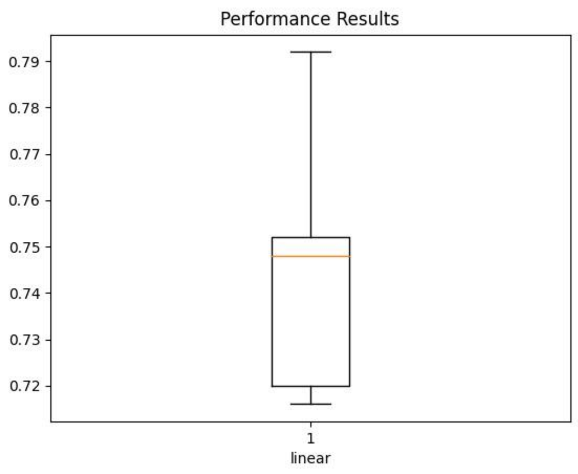

Figure 14: Performance results box plot of linear kernel model

Precision: 0.752 C: 0.01

Precision: 0.72 C: 0.01

Precision: 0.792 C: 0.01

Precision: 0.748 C: 0.01

Precision: 0.716 C: 0.01

Question 4.2

Grid search is performed on the hyper-parameter space and according to validation set, hyper-parameter C and γ is tuned and $C = 10, \gamma = scale$ was selected as the best model for this specific shuffled dataset. It takes approximately 400 seconds to run the setup for five different test set. Precision results' boxplot can be seen in the Figure 15 and precision results for each test fold can be seen below.

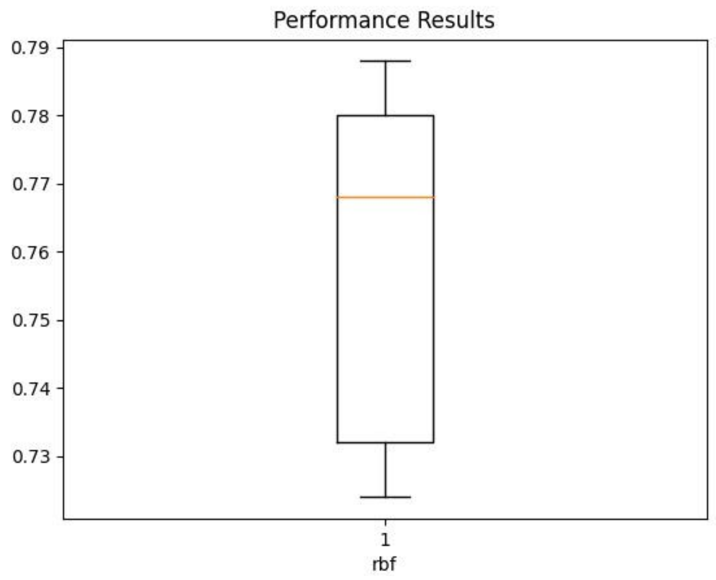

Figure 15: Performance results box plot of RBF kernel model

Precision: 0.78 C: 10 gamma: scale

Precision: 0.724 C: 10 gamma: scale

Precision: 0.768 C: 10 gamma: scale

Precision: 0.788 C: 10 gamma: scale

Precision: 0.732 C: 10 gamma: scale

Question 4.3

Test set performances of the models is shown in Figure 16.

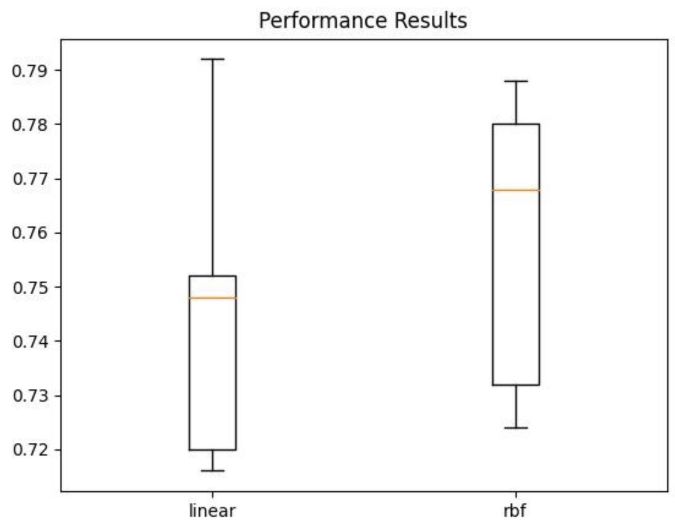

Figure 16: Performance results box plot of two kernel models

As it can be seen in Figure 16, kernel with radial basis function (RBF) gives better precision (approximately %2 to %4) results than linear kernel model in general. In RBF kernel model, C is mostly 10, but in linear model C is mostly 0.01. Since the precision does not increase much with RBF kernel, it could be said that the given data is almost linear. When $C = 10$, $\gamma = scale$ the best precision results are obtained and when $C = 10^{-6}$ for every γ value is the worst model. $C = 10^{-6}$ and different γ values gave similar precision with $C = 10^{-6}$ in linear kernel precision results. If SVM was trained on image pixels, first features should be calculated for randomly guessed weights which would give less accurate results than pre-trained features results. Therefore, a decrease would be expected for performance.

C, penalty parameter, represents the misclassification or error term. In other words, tells how much error is bearable to the SVM model. γ parameter decides how many data points can be considered for the decision boundary of SVM. If γ is increased too much, decision boundary of SVM would over fit.

Model will perform more accurately if hyper-parameter C applied without bounds. As C approaches infinity, this means that having any slack variable set to non-zero would have infinite penalty. Therefore, as C approaches infinity, decision boundary of SVM would be very sensitive to outliers and would be more likely to over fit.