CS464 - Introduction to Machine Learning

Homework Assignment - 2

Berkan Ozdamar

21602353

Contents

1 PCA on Van Gogh Paintings

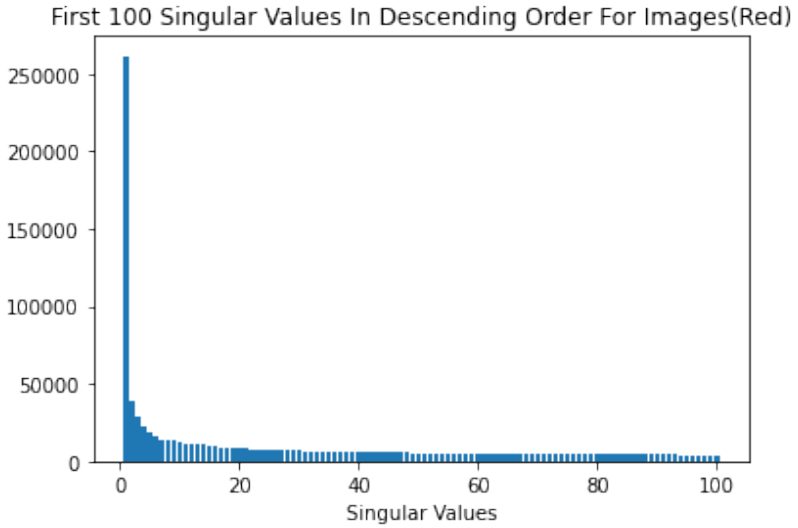
In this question, we are given 877 Van Gogh paintings. The images are preprocessed as asked in the question. First, grayscale images are stacked along the third dimension to obtain RGB version and then the images are flattened. Finally, R, G and B values are seperated so that can be considered alone.

Question 1.1

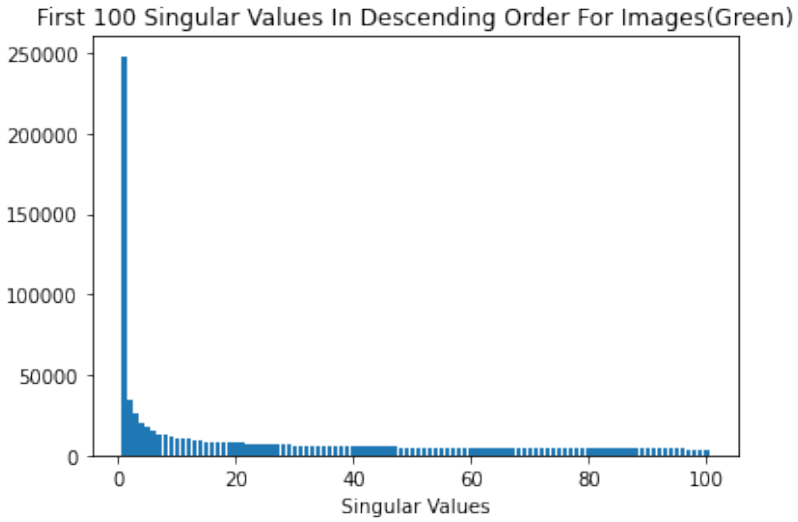
We are asked to apply SVD based approach to obtain first 100 principal components per color channel (R,G and B). Then the proportion of variance (PVE) explained by 10 principal components are asked to be found.

The proportion of variance will be computed using eigenvalues. We are computing SVD of the dataset, so the eigenvalues will be computed by the squares of the sigma values found from SVD. Then, we know the eigenvalue with the largest value is the first principal component.

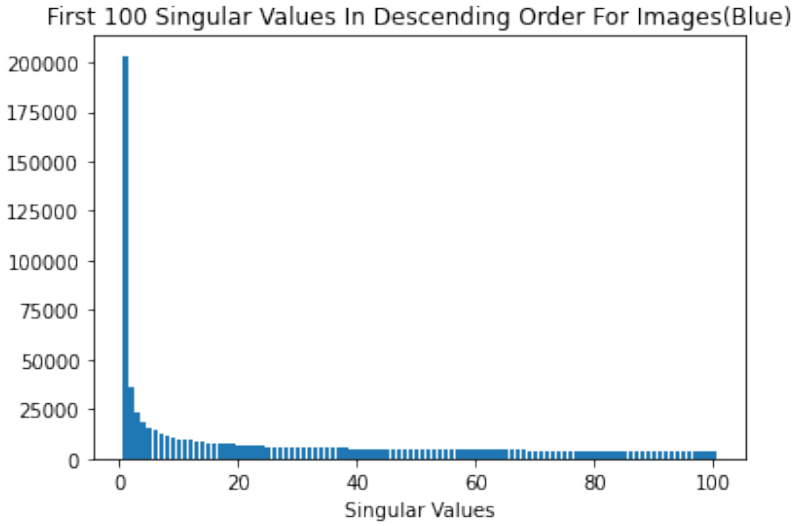
The first 100 principal components and the PVE explained by the first 10 principal components are given below.




```
Proportion of variance explained (PVE) by the first 10 principal components
(Red): 0.9108462700160358
```



```
Proportion of variance explained (PVE) by the first 10 principal components
(Green): 0.9090016283398422
```



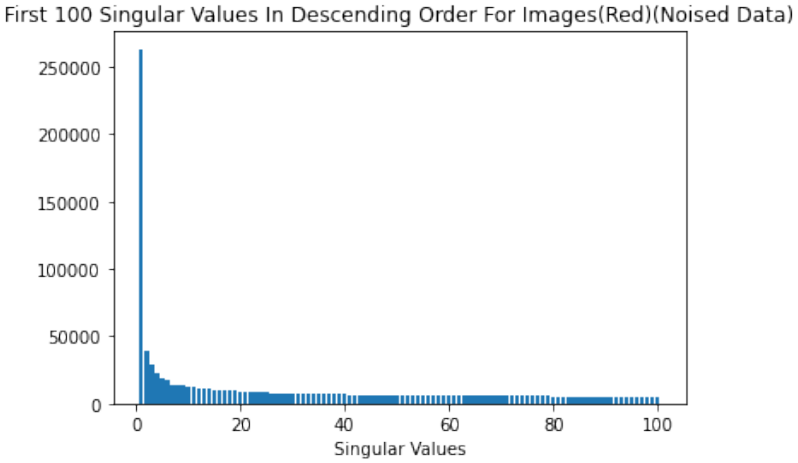

```
Proportion of variance explained (PVE) by the first 10 principal components
(Blue): 0.8831153502101908
```

From the figures of R,G and B values, it can be seen that the first principal component explains the most of the proportion of variance (PVE) for each of them. And the first 10 principal components explained roughly around 88% to 91% variance of the dataset. That means with only 10 principal components, 88% to 91% of variance is explained even though the feature size is reduced significantly.

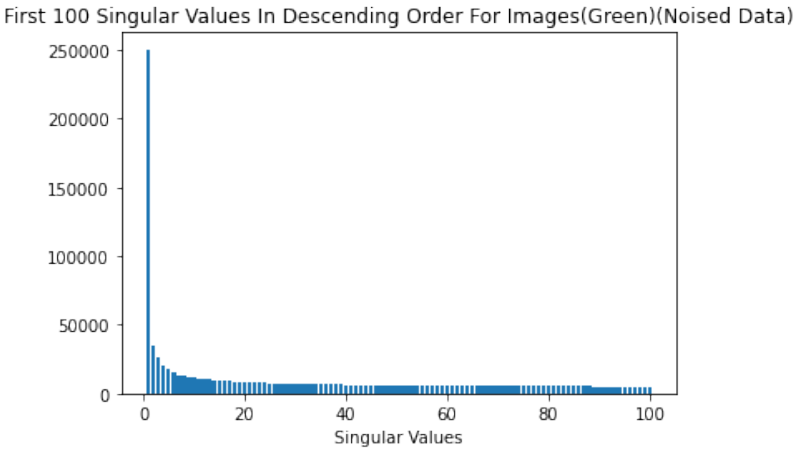
Question 1.2

In this question, a gaussian noise with mean μ and variance σ^2 will be added to the dataset. The mean μ and variance σ^2 are the mean and variance of the original dataset. However, noise will be scaled with a factor of 0.01 before added to the images. Then the first 100 principal components and the PVE explained by the first 10 principal components will be calculated again.

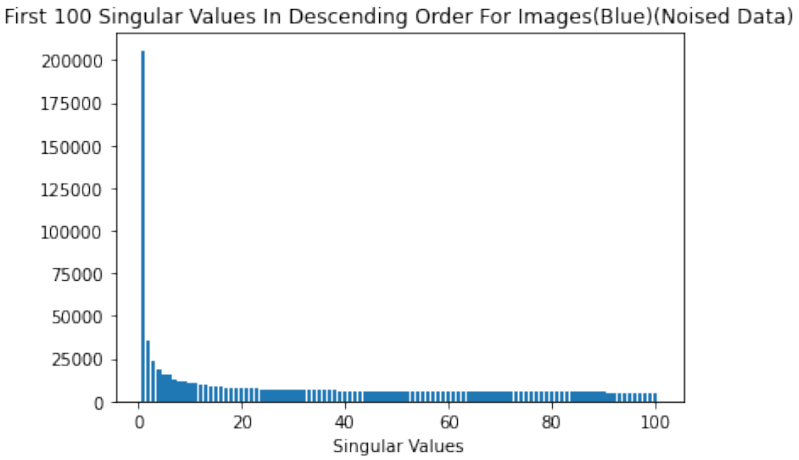
The first 100 principal component and the PVE explained by the first 10 principal components of the noisy data is given below:



```
Proportion of variance explained (PVE) by the first 10 principal components
(Red)(Noised Data): 0.8481034141795335
```




```
Proportion of variance explained (PVE) by the first 10 principal components
(Green)(Noised Data): 0.8396852542292162
```




```
Proportion of variance explained (PVE) by the first 10 principal components
(Blue)(Noised Data): 0.7741740290070318
```

- As can be seen from the results, the noise affected the PVE by first 10 principal components significantly. The variance explained by the first 10 principal components are much lower than the data with no noise applied. Because of the noise, the higher number of principal components have larger singular values, thus the PVE is decreased for first 10 principal components.
- The reconstruction of the image can be done with the $image_{reconstructed} = U_{(64,k)} S_{(k,k)} V^T_{(k,64)}$ where the k is the number of principal components that will be used for reconstruction of the image.

2 Linear Regression on University Admission Records

In this question, we are given University Admission Records of 500 students and a regression model will be trained on the dataset to predict the admission chance of a student. The features of the dataset are, GRE Score, TOEFL Score, University rating, Statement of Purpose, Letter of Recommendation, CGPA and Research.

Question 2.1

We are asked to derive the closed form solution for multivariate regression model using ordinary least squares(OLS) as the loss function. We can write the (OLS) as:

$$J_n = ||y - X\beta||^2 = (y - X\beta)^T(y - X\beta)$$

For the closed form solution, we want to take the partial derivative of OLS with respect to weights β and set it equal to 0. Thus, the equation is:

$$\frac{\partial J_n}{\partial \beta} = -2X^T(y - X\beta) = 0$$

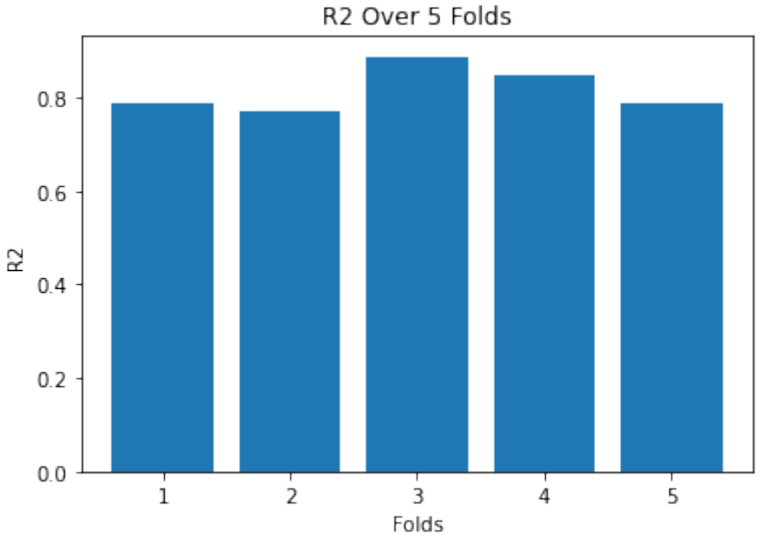
$$\frac{\partial J_n}{\partial \beta} = -X^T y - X^T X\beta = 0$$

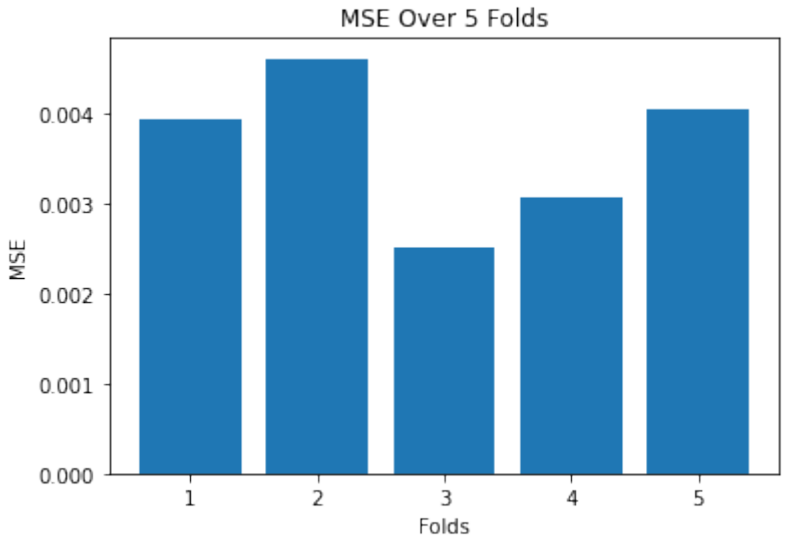
$$\beta = (X^T X)^{-1} X^T y$$

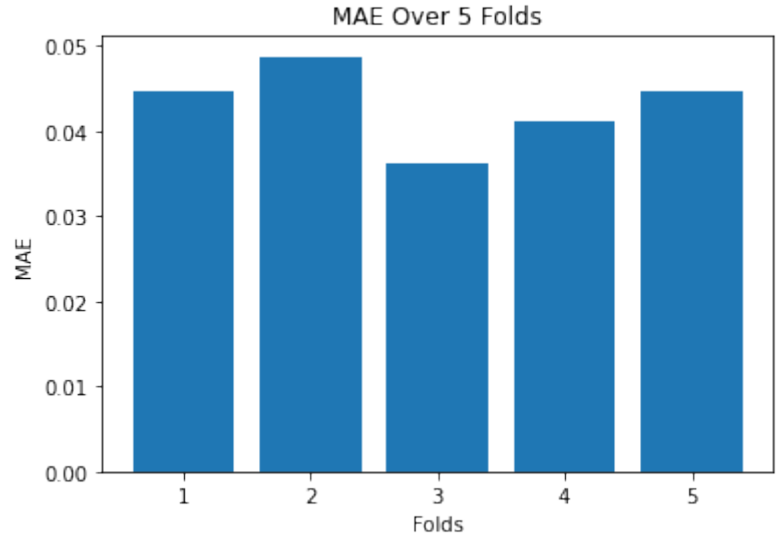
Question 2.2

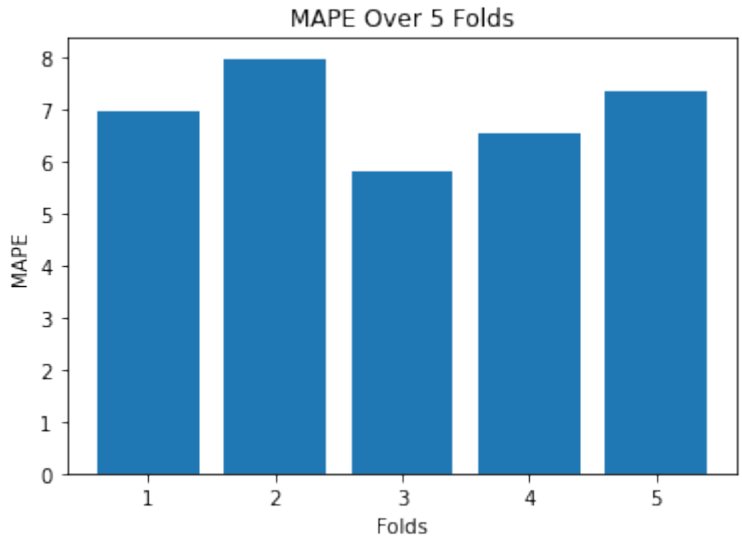
In this question, we are asked to train the linear regression model using the closed form solution for multivariate regression model we have found in *Question 2.1*. The 5-fold cross validation will be implemented and for each model, R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) will be calculated.

The R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) of each fold is given below:




MSE Over 5 Folds
MSE
0.000
0.001
0.002
0.003
0.004
1
2
3
4
5
Folds


MAE Over 5 Folds
MAE
0.00
0.01
0.02
0.03
0.04
0.05
1
2
3
4
5
Folds



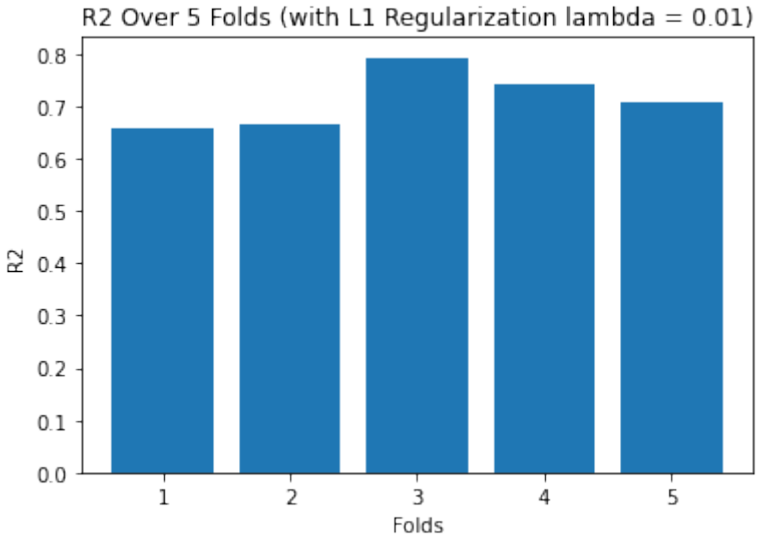
The mean of R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) for 5 folds are as:

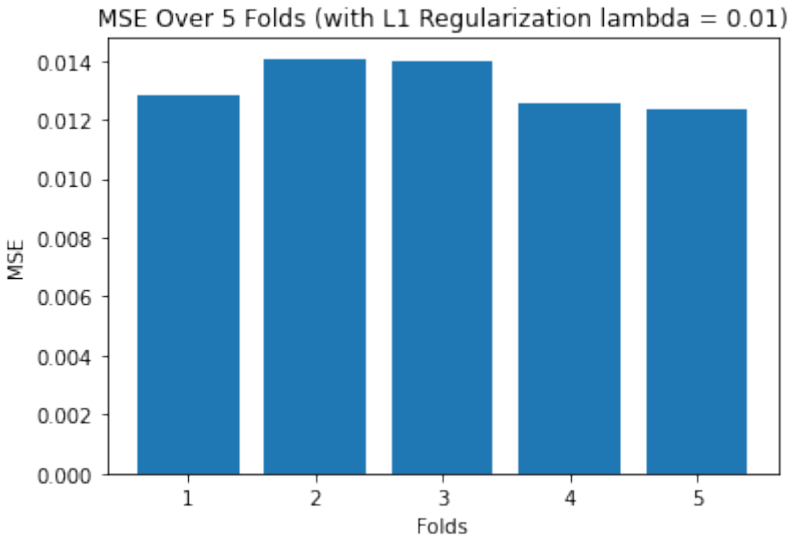
```
The mean of the R2 for 5 folds is   : 0.8160459106453812
The mean of the MSE for 5 folds is  : 0.0036271929106424946
The mean of the MAE for 5 folds is  : 0.04307313911239631
The mean of the MAPE for 5 folds is : 6.937989555383938
```

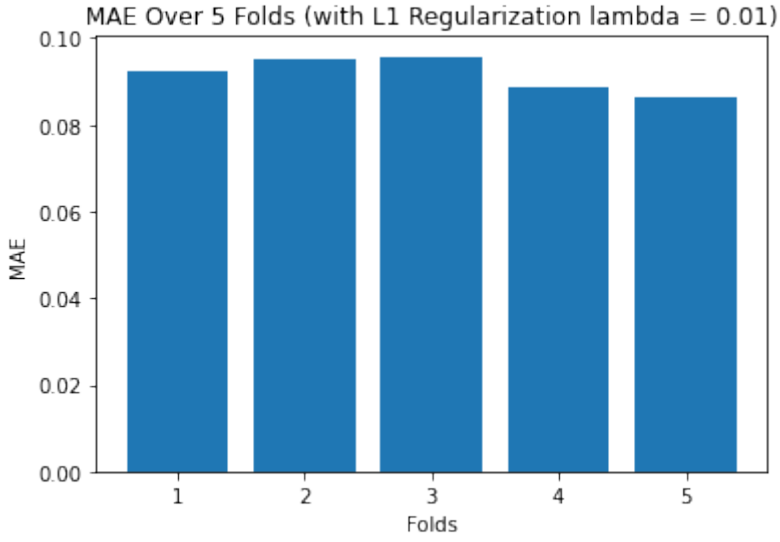
Question 2.3

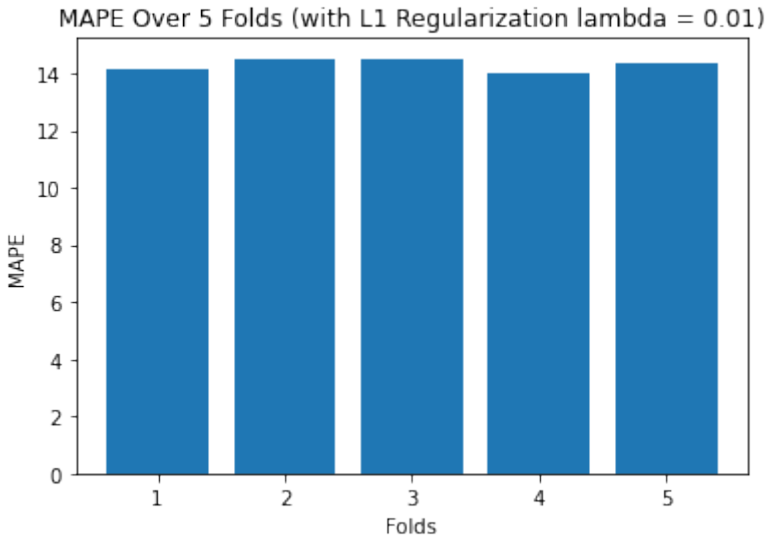
In this question, we will train the linear regression model with L1 regularization (Lasso) with $\lambda = 0.01$ and $\lambda = 1$. For the implementation of Lasso, gradient descent with 1000 epochs is done.

The R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) values for the model trained with $\lambda = 0.01$ is given below:




MSE Over 5 Folds (with L1 Regularization lambda = 0.01)
MSE
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
1
2
3
4
5
Folds

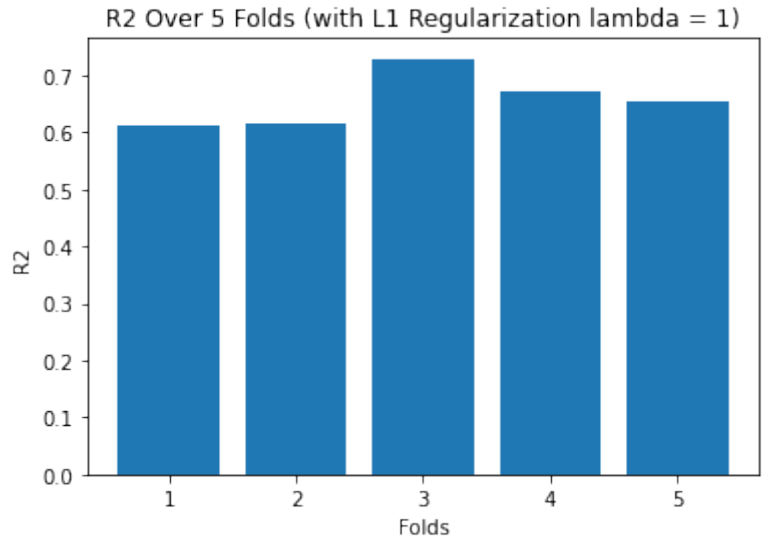

MAE Over 5 Folds (with L1 Regularization lambda = 0.01)
MAE
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
Folds

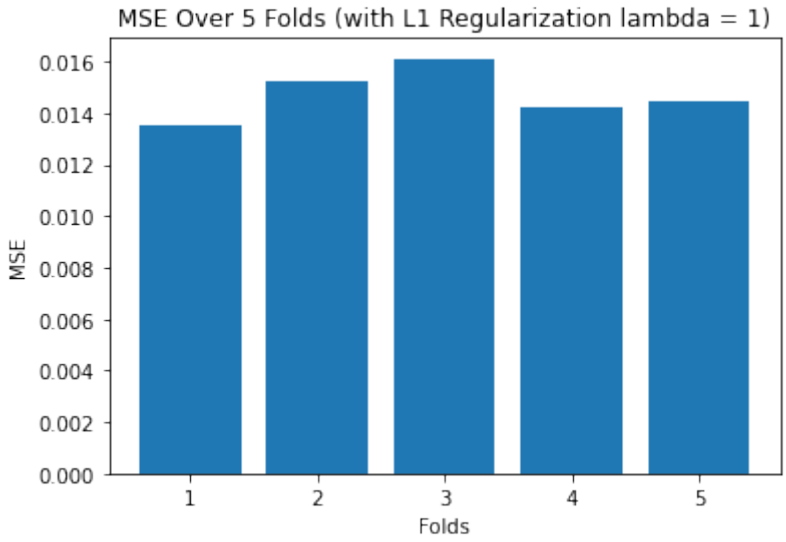


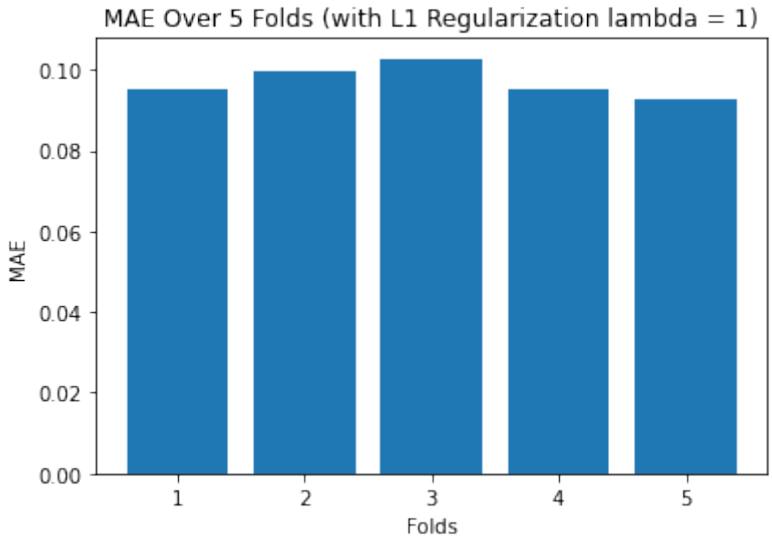
The mean of R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) values for the model trained with $\lambda = 0.01$ is given below:

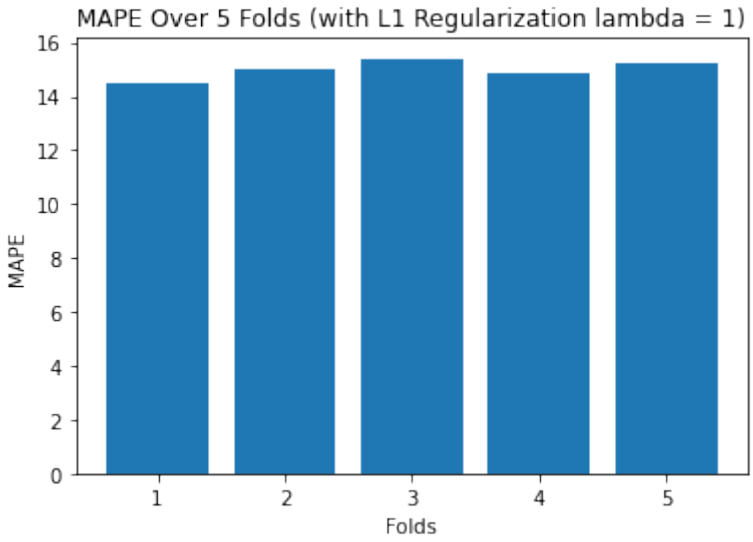
```
The mean of the R2 for 5 folds is (L1 with 0.01 lambda)    : 0.7123804612893659
The mean of the MSE for 5 folds is (L1 with 0.01 lambda)   : 0.013149189112525914
The mean of the MAE for 5 folds is (L1 with 0.01 lambda)   : 0.09159346873139199
The mean of the MAPE for 5 folds is(L1 with 0.01 lambda)   : 14.281201451863623
```

The R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) values for the model trained with $\lambda = 1$ is given below:




MSE Over 5 Folds (with L1 Regularization lambda = 1)
MSE
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
1
2
3
4
5
Folds


MAE Over 5 Folds (with L1 Regularization lambda = 1)
MAE
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
Folds



The mean of R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) values for the model trained with $\lambda = 1$ is given below:

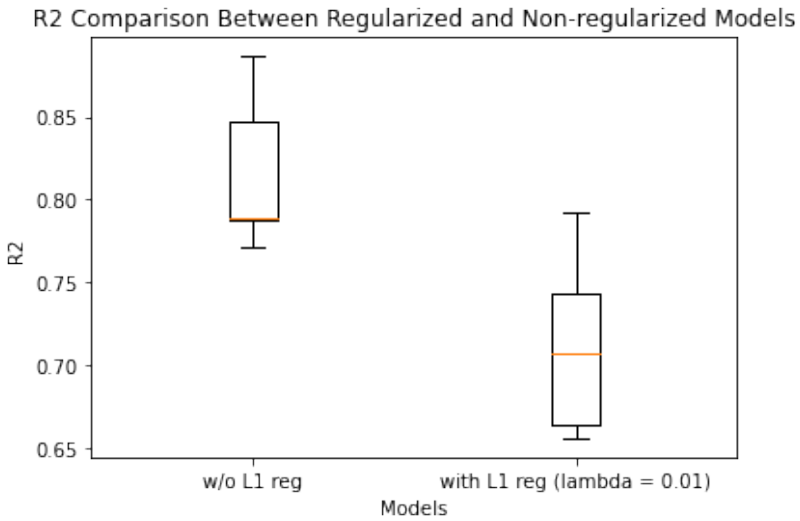
```
The mean of the R2 for 5 folds is (L1 with 1 lambda)    : 0.6567417349745028
The mean of the MSE for 5 folds is (L1 with 1 lambda)   : 0.014717510409331485
The mean of the MAE for 5 folds is (L1 with 1 lambda)   : 0.09698451402799471
The mean of the MAPE for 5 folds is(L1 with 1 lambda)   : 14.991403602363835
```

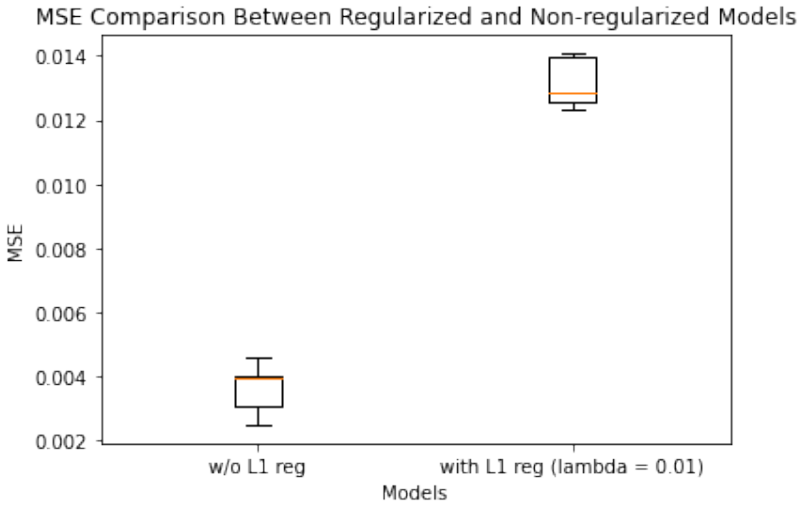
The performances of both of the models are similar but the model with $\lambda = 0.01$ is performing slightly better. Normally, we would expect the model with $\lambda = 0.01$ to perform better since the model with $\lambda = 1$ limits the weights too much and derives more zeros. If the regularization strength increases too much, then the penalty grows and coefficients will approach zero. Or if the regularizaton strength is too low, then the penalty is shrinks and may even have small to no effect on the network.

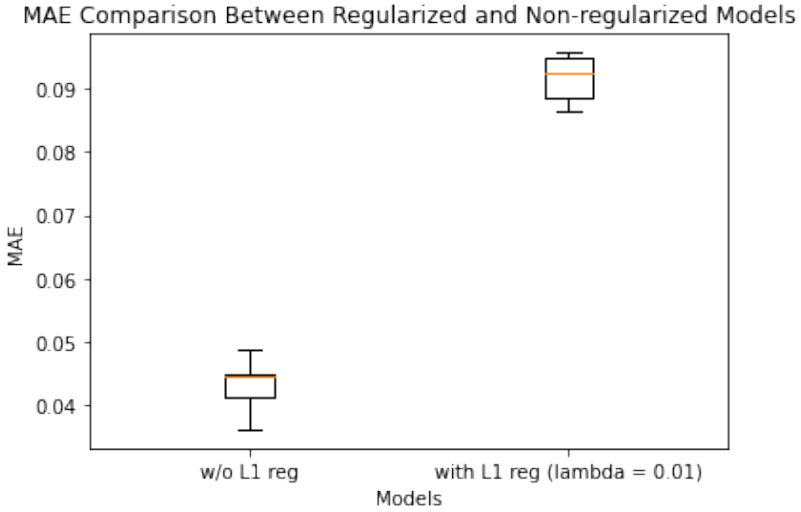
Question 2.4

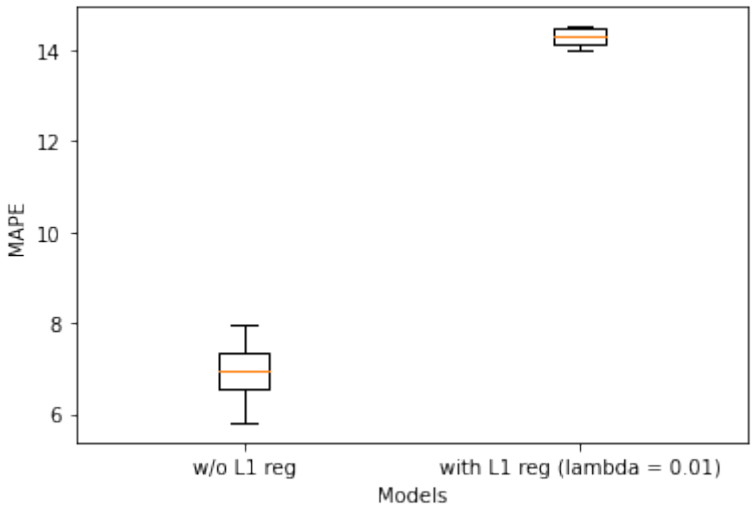
In this question, we will compare the trained linear model in *Question 2.2* and *Question 2.3*. For the regularized model in *Question 2.3*, I have used the model with $\lambda = 0.01$.

The comparisons of both models with respect to R^2, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) are shown below:

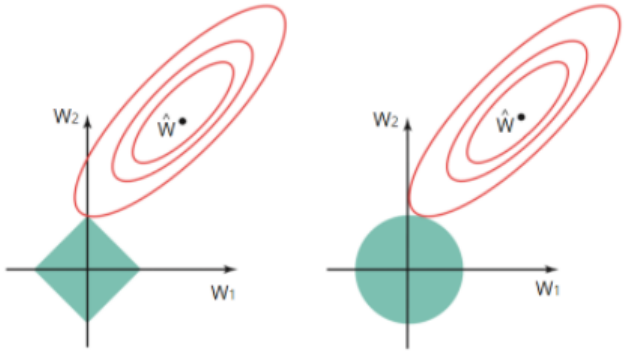



MSE Comparison Between Regularized and Non-regularized Models
0.014
0.012
0.010
0.008
0.006
0.004
0.002
MSE
w/o L1 reg
with L1 reg (lambda = 0.01)
Models


MAE Comparison Between Regularized and Non-regularized Models
0.09
0.08
0.07
0.06
0.05
0.04
MAE
w/o L1 reg
with L1 reg (lambda = 0.01)
Models


14
12
10
8
6
MAPE
w/o L1 reg
with L1 reg (lambda = 0.01)
Models

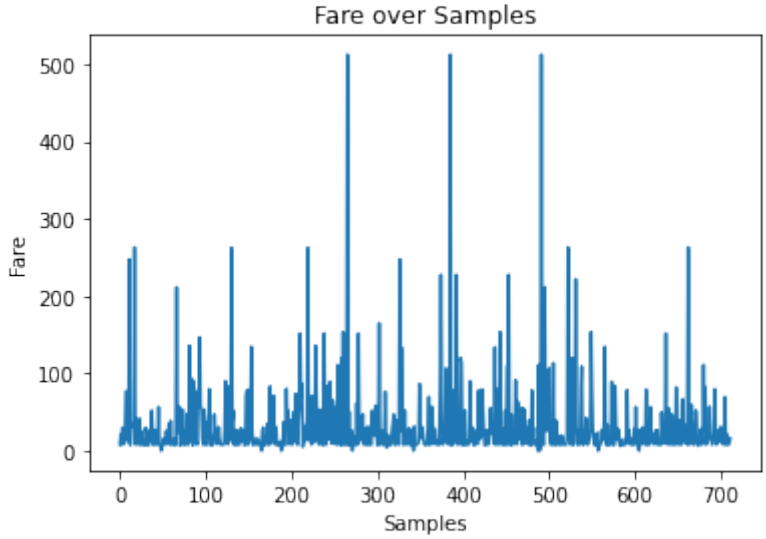
- Outliers are observations that are much smaller or much larger than the majority of the observations. Thus, outliers may effect the mean performance or they can increase the variability. Between MSE and MAE, MAE fails to punish large errors. Since MSE squares up the error, the outliers are more considered. That is why the MSE performances are similar between two models.
- The MSE would be my choice of the single error metric. MSE is usually optimizes better for in gradient descent algorithms. This could be seen from the boxplot comparisons of the regularized and non-regularized models.
- Since the given dataset size is very small, the model could not train well with a fixed dataset. Thus, to increase complexity of the trained model, cross validation is used for model to learn better with the small sized dataset.
- L1 regularization reduces the variance by penalizing the flexibility of the model. This approach avoids overfitting.

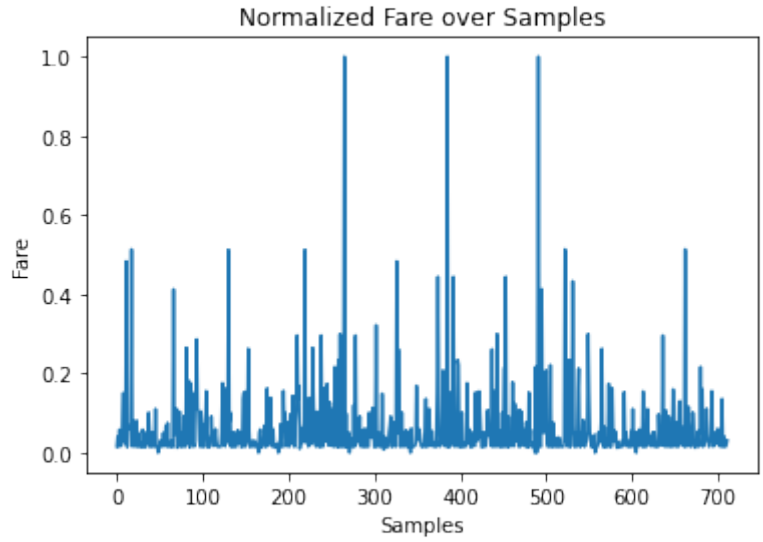


  However, since the L1 has diamond shape constraints, the intersections will occur more often with contours of loss function than L2. Thus, L1 encourages the weights to become zero more than L2. This can effect the performance of the model. Also, if the features are correlated, the model with L1 would underfit and perform poorly.
- If we would have a dataset size of 50000 instead of 500, then I would prefer a fixed test dataset approach since the dataset size is big enough to train it with fixed test dataset. The cross validation might be better performance wise, but the computational load is too much and probably unnecessary for such a big dataset.

3 Logistic Regression for Survival Prediction

In this question, we are given passenger information of the Titanic disaster and we are asked to predict the passenger survival using a logistic regression model. First, the categorical features are transformed to one-hot encoding features. The features encoded with one hot encoding are gender and port of embarkation. Instead of labeling, I have used one hot encoding since there is no hierarchical order between the values of these features so independency is protected with one hot encoding. Then, the features are normalized since the range of different features range in different intervals and this can effect the performance of the logistic regression. An example of the feature *Fare*, before and after normalization is given below:





Thus, all the features are normalized into interval (0,1).

Question 3.1

A mini-batch gradient ascent with batch size = 32 and a stochastic gradient ascent algorithm is implemented in this part. All the weights are initialized with random numbers from a Gaussian distribution

with mean 0 and standard deviation (0.01). 1000 iterations are performed. The learning rate is optimized as 10^{-2} for mini-batch gradient ascent and $10^{-4}$for stochastic gradient ascent model. Confusion matrix, accuracy, precision, recall, negative predictive value, false positive rate, false discovery rate, F1 and F2 scores are given below for both mini-batch gradient ascent and stochastic gradient ascent:

```
Confusion Matrix for mini-batch gradient ascent


True Positive : 48
True Negative : 101
False Positive : 9
False Negative : 21
----------------------------------

Scores for Mini-Batch Gradient Ascent


Accuracy                  : 0.8324022346368715
Precision                 : 0.8421052631578947
Recall                    : 0.6956521739130435
Negative Predictive Value : 0.8278688524590164
False Positive Rate       : 0.08181818181818182
False Discovery Rate      : 0.15789473684210525
F1 Score                  : 0.761904761904762
F2 Score                  : 0.7207207207207208
----------------------------------

Confusion Matrix for stochastic gradient ascent


True Positive : 35
True Negative : 73
False Positive : 37
False Negative : 34
----------------------------------

Scores for Stochastic Gradient Ascent


Accuracy                  : 0.6033519553072626
Precision                 : 0.4861111111111111
Recall                    : 0.5072463768115942
Negative Predictive Value : 0.6822429906542056
False Positive Rate       : 0.33636363636363636
False Discovery Rate      : 0.5138888888888888
F1 Score                  : 0.49645390070921985
F2 Score                  : 0.5028735632183908
----------------------------------
```

From the confusion matrices and performance metrics, it can be seen that the mini-batch gradient ascent performed better than stochastic gradient ascent. SGD takes a sample and updates a weights for all samples in one epoch. This makes the model to fluctuate while converging since single samples may be noisy. However, mini-batch gradient ascent takes a batch from the whole sample and computes the gradient from that batch then update the weights. The mini-batches averages the noise, that creates much less fluctuations while training.

Question 3.2

In this question, a full-batch gradient ascent is implemented. All the weights are again initialized with random numbers from a Gaussian distribution with mean 0 and standard deviation (0.01). The learning rate is optimized as 10^{-3}. 1000 iterations are performed. Confusion matrix, accuracy, precision, recall,

negative predictive value, false positive rate, false discovery rate, F1 and F2 scores are given below for the full-batch gradient ascent:

```
Confusion Matrix for full-batch gradient ascent


True Positive : 44
True Negative : 100
False Positive : 10
False Negative : 25
----------------------------------

Scores for Full-Batch Gradient Ascent


Accuracy                  : 0.8044692737430168
Precision                 : 0.8148148148148148
Recall                    : 0.6376811594202898
Negative Predictive Value : 0.8
False Positive Rate       : 0.09090909090909091
False Discovery Rate      : 0.18518518518518517
F1 Score                  : 0.7154471544715448
F2 Score                  : 0.6666666666666666


Weights at each 100th iteration:


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
```

```
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


[ 6.74609721e-03 -7.00551816e-04 -6.22274797e-04 -1.23936059e-04
  7.99398536e-05 -1.29297963e-03 -6.26480021e-03 -5.70209562e-03
 -6.15319822e-05  1.00999427e-04  3.47849519e-05]


---------------------------------
```

It can be seen from the weights updates per 100 iterations, the weights for full-batch gradient ascent does not update the weights after the 100th iterations. That means that the full batch gradient ascent is most probably stuck on a local maximum. Since the full batch gradient ascent is updating weights after an epoch, it is harder for full batch to get out of the local maximum than the mini-batch and stochastic gradient ascent.

Question 3.3

- The feature importance can be observed from the values of weights of that corresponding feature if all the features are normalized. If the weights have high values for that feature, that means that feature is important in our classification problem. In our case, the gender data is 10 to 20 times larger than the most of the features, thus we can infer that the gender feature is important for the survival classification.
- The normalization is important for feature importance since normalization scales all the features to the interval (0,1). This is particularly important when some features have too big or too small values like Fare feature in our example. Without normalization, since Fare values are high, it would put emphasis on the model even if it is not actually important.
- It is possible to compare the importance of the categorical and continuous features together since we have applied one hot encoding to the categorical features. Since, with one hot encoding, the categorical features are converted, then the weights of each feature shows the importance of the each feature. From looking at the weights in the *Question 3.2*, gender and ticket class are the most important features.
- When the batch size of the mini-batch increases, there will be less weight updates, thus the time of the training will be much less. As batch size decreases, the computation need increases and the time of the training increases as well.

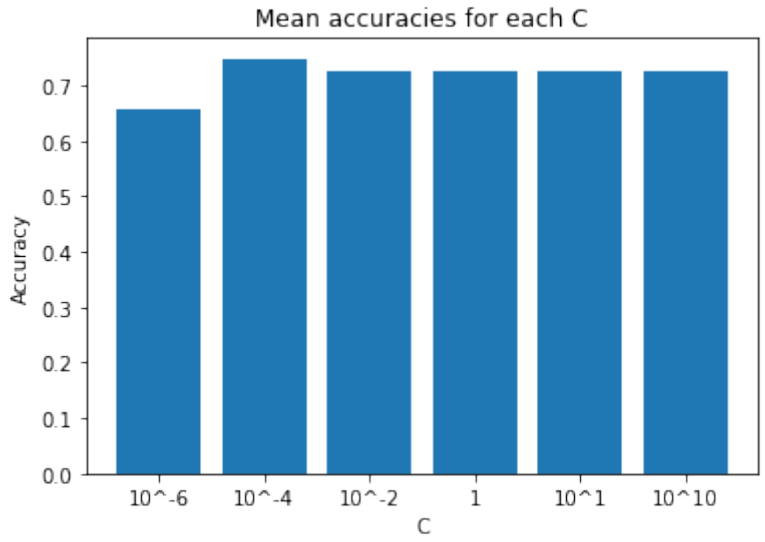
4 SVM

Question 4.1

In this question, we are asked to classify flowers using multi-class SVM models with one-vs-all approach. Also, we are given Inception v3 which is a pretrained deep convolutional neural network which is trained with the ImageNet dataset.

First, SVM with Linear Kernel will be trained with a stratified 5-fold cross validation with train, validation and test data. The hyperparameter turning will be done by the accuracy of the validation data. The best hyperparameters and the time of training for the Linear SVM is given below:

```
Best parameters for SVM (linear) : {'best_c': 0.0001, 'best_gamma': None}


The scores for SVM (linear) : [0.6567999999999999, 0.748, 0.7256, 0.7256,
0.7256, 0.7256]


Time elapsed on training SVM (linear) : 116.19397521018982 seconds
```



Then, the Linear SVM will be trained with stratified 5-fold cross validation with only training and test dataset with the hyperparameters found above. The classification reports of the Linear SVM for 5 folds is given below:

```
SVM(linear) with best parameters:


Fold 1:
              precision    recall  f1-score   support

           0       0.81      0.84      0.82        56
           1       0.87      0.77      0.82        53
           2       0.74      0.71      0.73        49
           3       0.67      0.80      0.73        41
           4       0.71      0.69      0.70        51

    accuracy                           0.76       250
   macro avg       0.76      0.76      0.76       250
weighted avg       0.77      0.76      0.76       250
```

Fold 2:

	precision	recall	f1-score	support
0	0.89	0.67	0.77	49
1	0.75	0.73	0.74	52
2	0.79	0.81	0.80	52
3	0.84	0.78	0.81	55
4	0.67	0.93	0.78	42
accuracy			0.78	250
macro avg	0.79	0.78	0.78	250
weighted avg	0.79	0.78	0.78	250

Fold 3:

	precision	recall	f1-score	support
0	0.84	0.79	0.81	47
1	0.80	0.76	0.78	46
2	0.70	0.83	0.76	47
3	0.83	0.68	0.75	57
4	0.69	0.77	0.73	53
accuracy			0.76	250
macro avg	0.77	0.77	0.77	250
weighted avg	0.77	0.76	0.76	250

Fold 4:

	precision	recall	f1-score	support
0	0.82	0.87	0.85	47
1	0.85	0.72	0.78	47
2	0.75	0.73	0.74	55
3	0.67	0.84	0.75	44
4	0.79	0.72	0.75	57
accuracy			0.77	250
macro avg	0.78	0.78	0.77	250
weighted avg	0.78	0.77	0.77	250

Fold 5:

	precision	recall	f1-score	support
0	0.88	0.88	0.88	51
1	0.81	0.65	0.72	52
2	0.64	0.79	0.70	47
3	0.64	0.68	0.66	53
4	0.65	0.60	0.62	47
accuracy			0.72	250

```
   macro avg       0.72      0.72      0.72       250
weighted avg       0.73      0.72      0.72       250
```

Question 4.2

The same procedure we have done in the *Question 4.1* will be done here. However, this time instead of Linear SVM, we will train a SVM with RBF Kernel. The hyperparameter tuning and testing procedure is the same before. The best hyperparameters for the SVM with RBF Kernel is given below:

```
Best parameters for SVM (rbf) : {'best_c': 10, 'best_gamma':
0.00019630923305203346}


Time elapsed on training SVM (rbf) : 1331.402500629425 seconds
```

Then the classification reports of the SVM with RBF Kernel for 5 folds is given below:

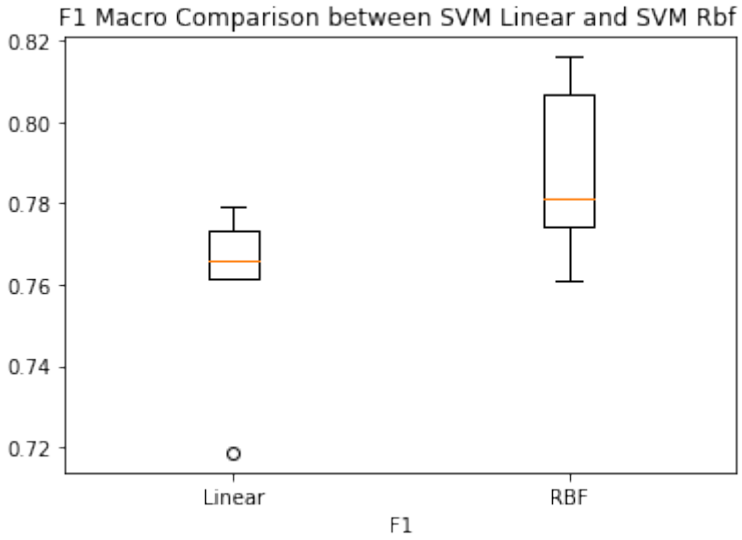
```
SVM(rbf) with best parameters:


Fold 1:
              precision    recall  f1-score   support

           0       0.84      0.84      0.84        56
           1       0.90      0.81      0.85        53
           2       0.74      0.69      0.72        49
           3       0.78      0.88      0.83        41
           4       0.78      0.82      0.80        51

    accuracy                           0.81       250
   macro avg       0.81      0.81      0.81       250
weighted avg       0.81      0.81      0.81       250



Fold 2:
              precision    recall  f1-score   support

           0       0.91      0.65      0.76        49
           1       0.78      0.77      0.78        52
           2       0.76      0.75      0.76        52
           3       0.82      0.84      0.83        55
           4       0.65      0.88      0.75        42

    accuracy                           0.78       250
   macro avg       0.79      0.78      0.77       250
weighted avg       0.79      0.78      0.78       250



Fold 3:
              precision    recall  f1-score   support

           0       0.86      0.81      0.84        47
           1       0.75      0.78      0.77        46
           2       0.70      0.83      0.76        47
           3       0.84      0.74      0.79        57
```

```
              4       0.77      0.75      0.76        53

    accuracy                            0.78       250
   macro avg       0.78      0.78      0.78       250
weighted avg       0.79      0.78      0.78       250
```

```
Fold 4:
              precision    recall  f1-score   support

           0       0.87      0.87      0.87        47
           1       0.84      0.77      0.80        47
           2       0.81      0.78      0.80        55
           3       0.77      0.82      0.79        44
           4       0.80      0.84      0.82        57

    accuracy                           0.82       250
   macro avg       0.82      0.82      0.82       250
weighted avg       0.82      0.82      0.82       250
```

```
Fold 5:
              precision    recall  f1-score   support

           0       0.98      0.84      0.91        51
           1       0.84      0.71      0.77        52
           2       0.64      0.79      0.70        47
           3       0.71      0.79      0.75        53
           4       0.69      0.66      0.67        47

    accuracy                           0.76       250
   macro avg       0.77      0.76      0.76       250
weighted avg       0.77      0.76      0.76       250
```

Question 4.3



- As can be seen from the boxplots, the SVM model with RBF kernel performs slightly better than the SVM model with Linear Kernel.
- If the SVM was trained directly on the image pixels, then the network would not perform better since the Inception v3 is a pre-manipulated dataset. In the original image, the irrelevant and noisy pixels are considered too, so definitely original image pixels need data manipulation before training with SVM too.
- C is the margin allowance parameter. The higher C value, the smaller margin hyperplane, so that the misclassification is restricted. When the c values are smaller, we have larger margin hyperplane so that the model lets some misclassifications. And gamma is the parameter that gives curvature to the decision boundary. The higher the gamma, the more curvature the decision boundary has.
- If we would have more training and test data, the model would perform better if the data is a relevant data.

References

[1] *Artem Oppermann,* "A Guide on the Theory and Practicality of the most important Regularization Techniques in Deep Learning" Available: https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout-377e75acc036

[2] "sklearn.multiclass.OneVsRestClassifier — scikit-learn 0.23.2 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html (accessed Dec. 09, 2020).