SPRING 2023-2024

CS 464: INTRODUCTION TO MACHINE LEARNING

Date: 14.04.2024

SECOND HOMEWORK REPORT

YİĞİT ALİ KARADOĞAN

21902218

DEPARTMENT OF INDUSTRIAL ENGINEERING

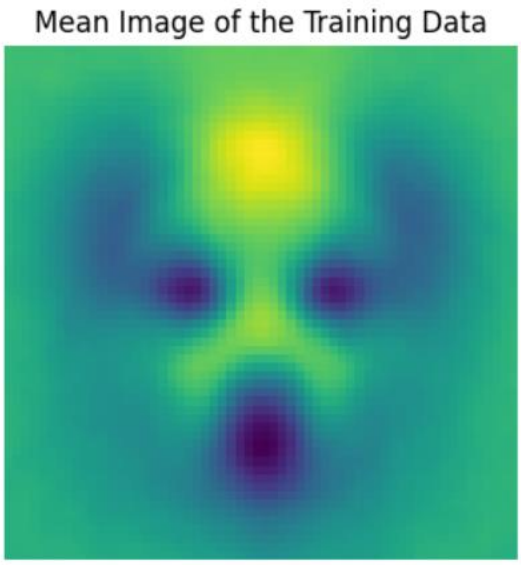
BILKENT UNIVERSITY

06800 ANKARA

Q1 – PCA

1-1

I answered first question by according to the instructions given. I initialized X matrix with 3000 rows and 3600 columns. The reason why there are 3600 columns is our images are 60x60. I flattened them and stored in X matrix. After that, I took the mean of the X matrix and subtracted from X to get centered X. It is needed for PCA calculations. I wanted to check the mean image of the data and it looks like the following image:

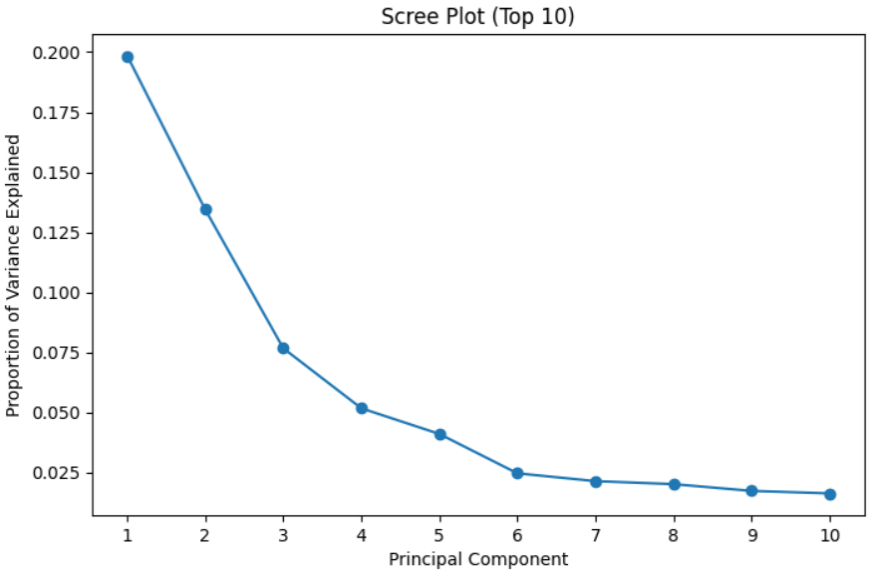


Prior to examining the mean figure, it makes sense to me that, as the mean will represent the assortment of dogs, I should anticipate seeing a fuzzy image that somewhat like the dog. As we look closely, we can notice the eyes, nose and the ears of a dog.

I computed the covariance matrix of X after centering it. The eigenvalues and eigenvectors of the matrix were then obtained. The eigenvectors corresponding to the 10 largest eigenvalues represent the first 10 principal components. The proportion of total variance explained by each principal component is equal to its corresponding eigenvalue divided by the sum of all eigenvalues. You can see top 10 principal components PVE and the total variance explained by them here:

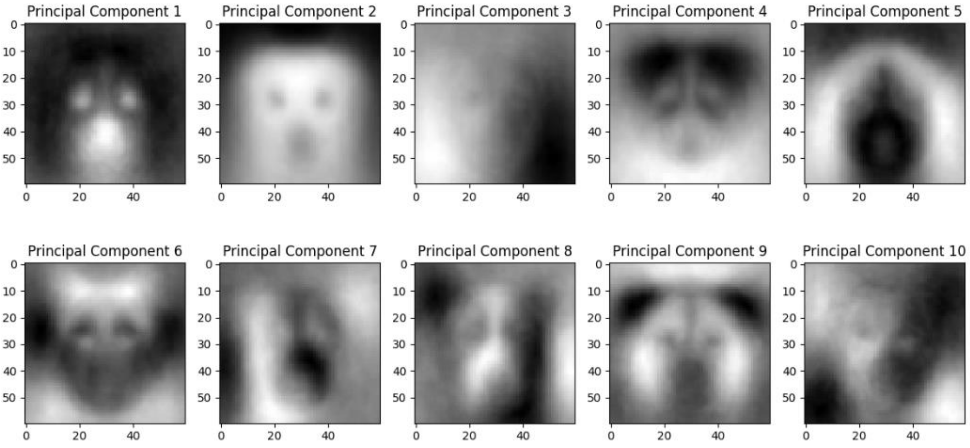
```
PVE for each principal component:
 [0.19841491 0.13446393 0.07677024 0.05181705 0.04113314 0.02476052
 0.02150967 0.02026394 0.01745666 0.01641553]
Total Variance explained by top 10 principal components:  0.603006
```

According to the analysis, the first principal component, which accounts for roughly 19.8% of the variance, captures the largest proportion of the variance. Subsequent principal components explain progressively smaller amounts of variance, with the second PC explaining 13.4%. This is because the first PC identifies the direction of greatest variability in the data. Total variance explained by top 10 principal components is nearly 60%. I plotted the scree plot of these 10 PCs, As anticipated, principal components explain progressively smaller amounts of variance:



By examining the PVE values of the principal components, we can determine how effective PCA is for dimensionality reduction. Since the first few principal components have high PVE values, it means that we can represent the data in a much lower-dimensional space with minimal information loss.

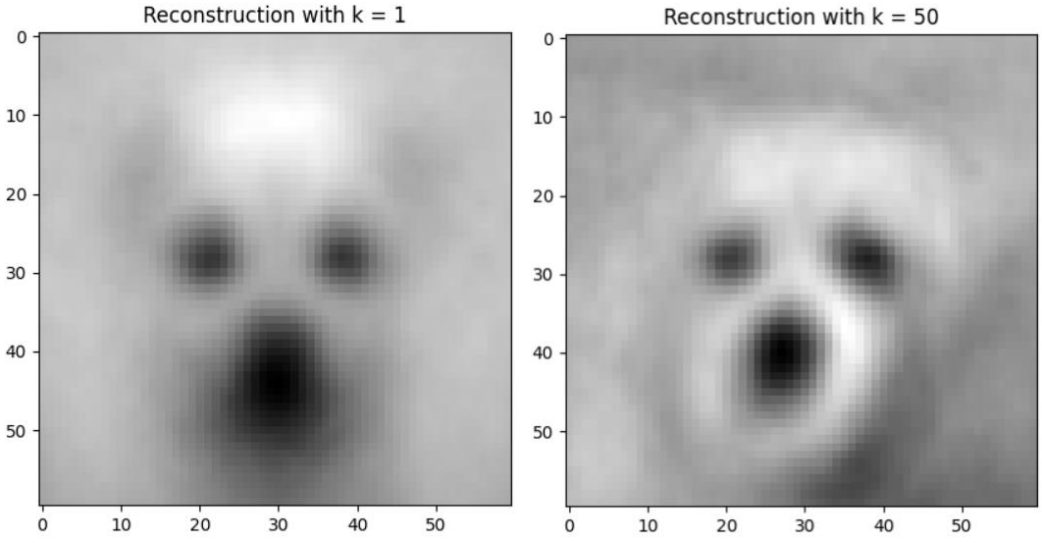
1-2

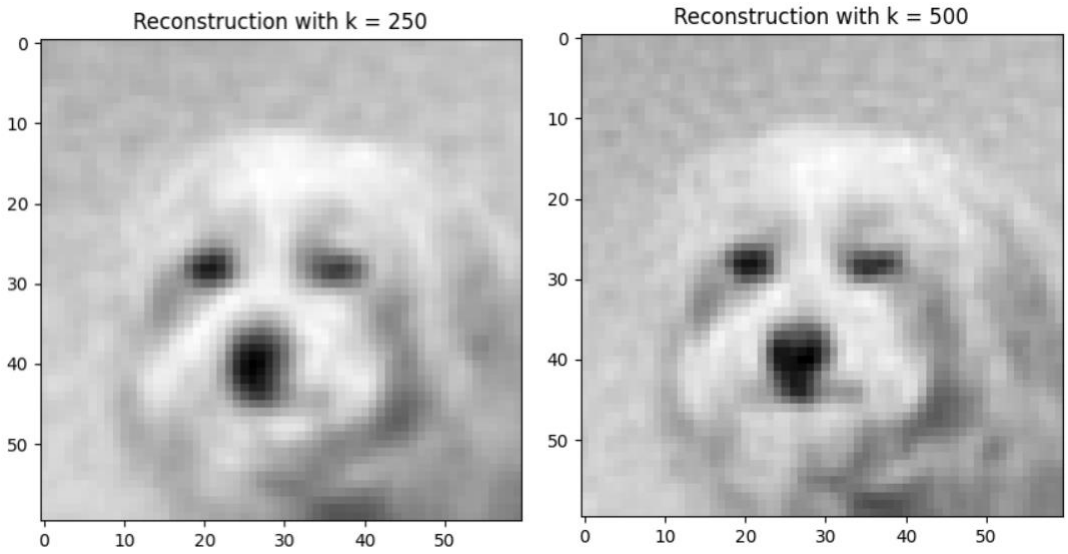


The most prevalent visual patterns in the dog images are represented by the first few principal components. These consist of the brightness of the image as a whole, basic outlines and shapes (such as the dog's silhouette), and typical pose variations. Since the first two principal components captures approximately one-third of the total variance, we can see a more general portre of a dog if we look at them. Later principal components tend to capture more smaller details, such as the dog's fur texture, tiny variations in features, like the shape of its ears, and maybe even noise. As anticipated, as we move forward, latter PC's captured more detailed patterns as the case in PC 6. They became less interpretable.

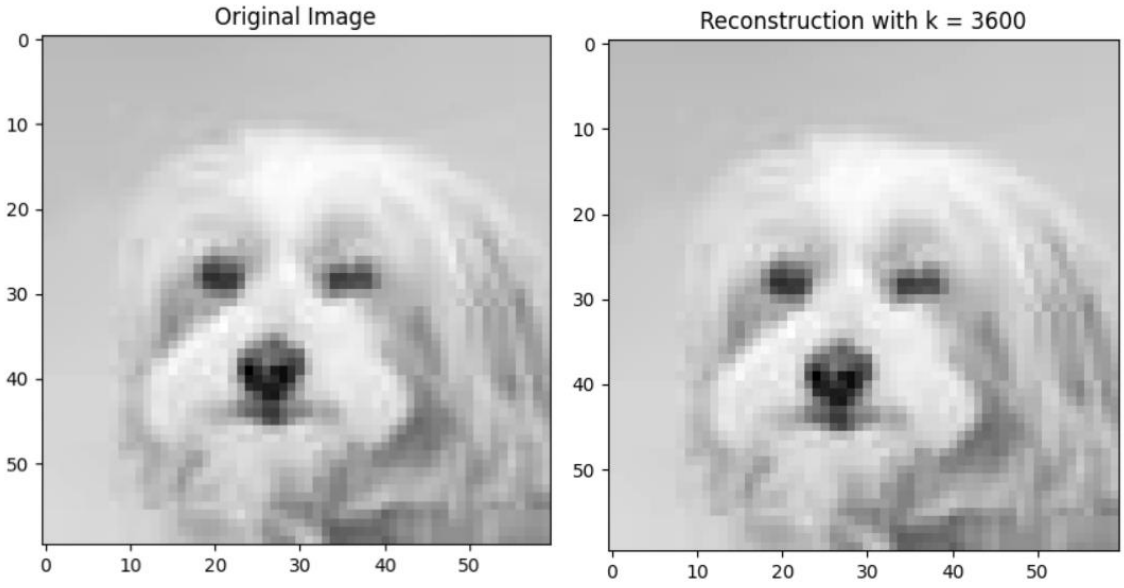
1-3

In order to create a lower-dimensional representation of a dog image, we project it onto a predetermined number of principal components (k) before reconstructing it using principal components. To produce a rough representation of the original image, this projection is inverted. Using a small k results in significant compression, as only the most dominant visual patterns from the dataset are retained, leading to a loss of detail in the reconstruction. More principal components are included as k increases, enhancing the reconstruction's quality and protecting the dog's finer features since we capture more variance as k increases. This illustrates how PCA-based image reconstruction balances image clarity and compression. The results with different k values are shown below:





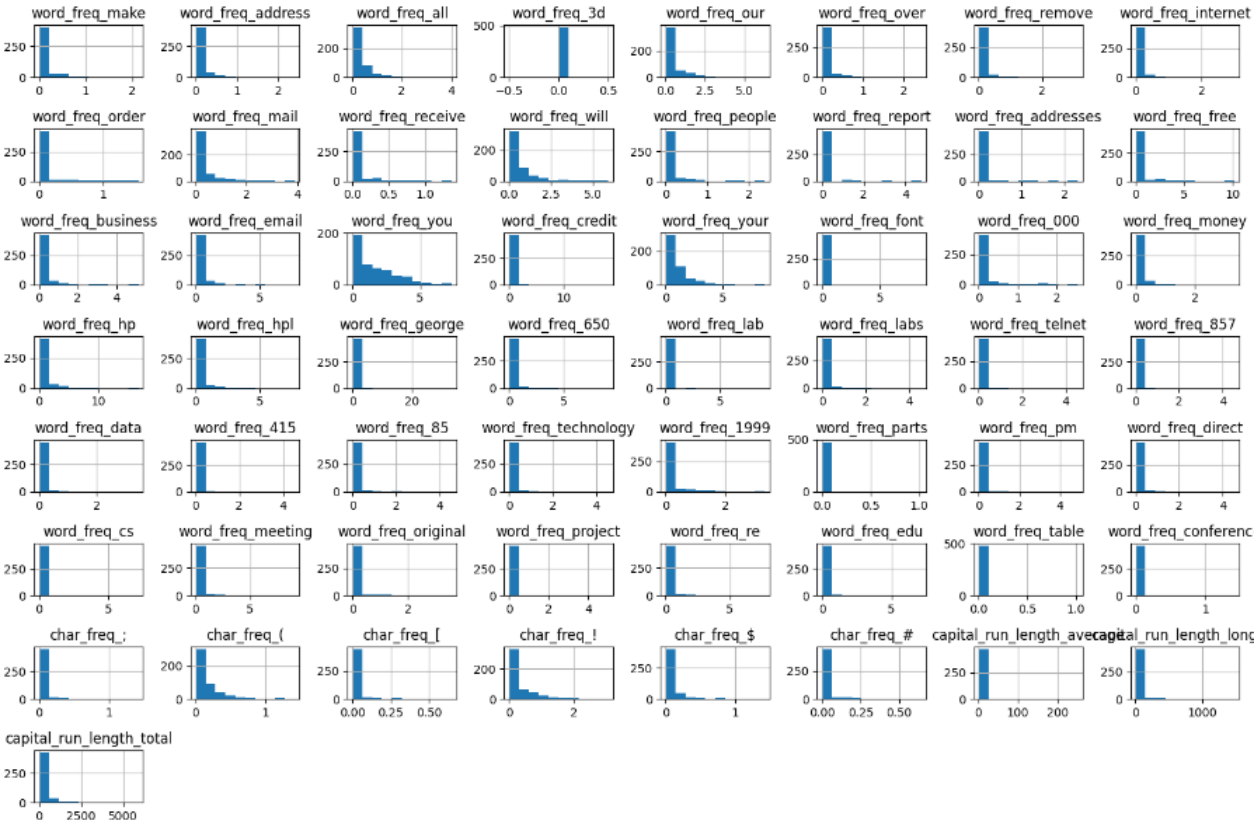
As anticipated, increasing the number of principal components used in the reconstruction results in a clearer and more detailed image of the dog. Note that even with the one principal component, dog's face is captured. Additionally, there is little to no difference between the original and reconstructed image for k = 3600 as you can see below.



Q2 – Logistic Regression

2-1

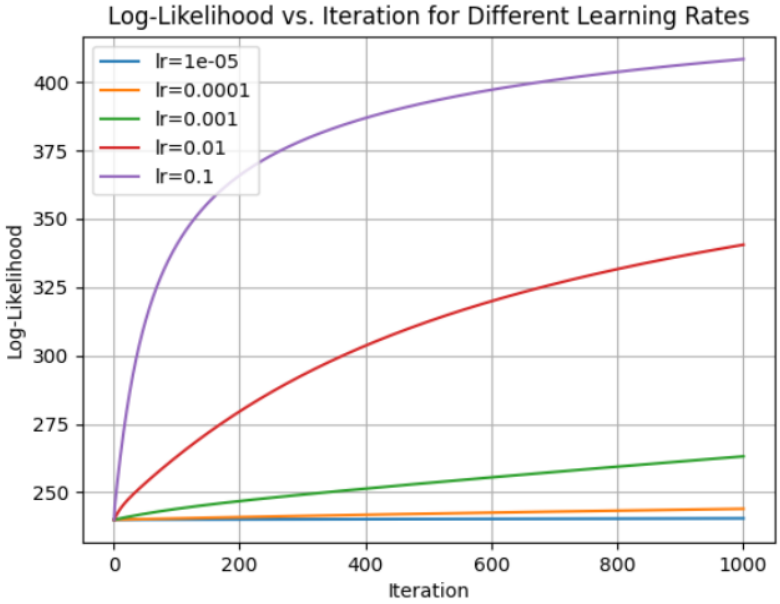
In this question, we are asked to use Spambase dataset provided. At first, I inspected the distributions of the features and plotted one by one. Here are the results:



Observing that the feature distributions were heavily right-skewed, I've implemented a preprocessing step to address this skewness. Log-transformation was applied to normalize the feature distributions. Before this step, I was encountering with the "overflow" problem for some of the learning rates. After this step, this issue has been solved.

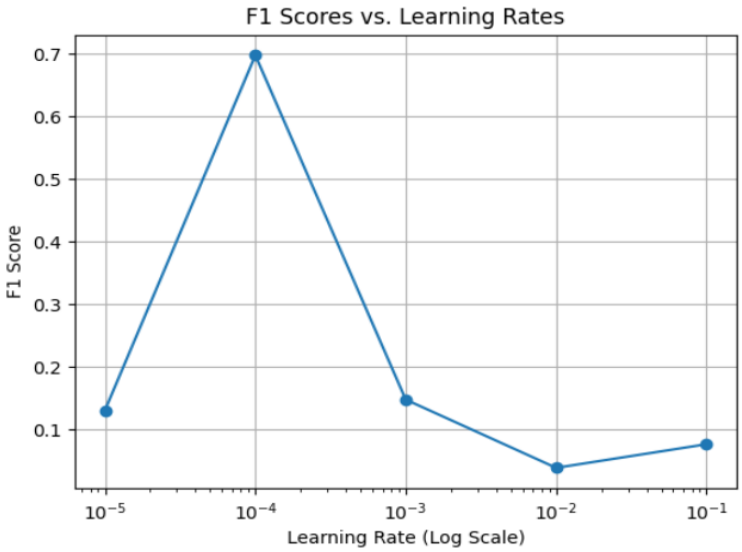
The gradient ascent step computes the error between the model's predictions and the true labels repeatedly, beginning with an initial set of weights. The direction of the largest increase is then indicated by computing the gradient of the log-likelihood function. A step in the gradient's direction, scaled by the learning rate, is taken to update the weights. The goal of this process is to converge on weights that maximize the log-likelihood and enhance the model's fit to the training set of data over a predetermined number of iterations. I expected log-likelihood increases are

different for different learning rates. For bigger learning rates, these jumps has to be more bigger. Hence, I plotted log-likelihoods for different learning rates:



Expectedly, higher learning rates at first cause log-likelihood to increase more quickly. Nevertheless, an unnecessarily high learning rate may result in erratic updates and hinder the model's convergence towards an optimal solution, thereby causing variations in the log-likelihood.

Crucially, good performance on unknown data does not always follow from a high log-likelihood on the training set. In this instance, the model with a learning rate of 10^{-4} demonstrated the best F1 score on the validation set despite achieving a lower log-likelihood on the training set. This observation draws attention to the possibility of overfitting. You can see F1 scores for different learning rates below:



Hence, I chose 10^{-4} as learning rate and decided to use for the test set. Note that since this is a binary classification problem, the micro-average and macro-average of precision will be the same. Here are the calculated values:

```
Confusion Matrix:
 [[44 19]
 [ 6 31]]
Performance Metrics:
Accuracy: 0.75
Precision: 0.70
Recall: 0.88
F1 Score: 0.78
F2 Score: 0.84
Negative Predictive Value (NPV): 0.84
False Positive Rate (FPR): 0.38
False Discovery Rate (FDR): 0.30
```

The model performs well as a spam filter, achieving an accuracy of 75%. It demonstrates good ability to catch spam (88% recall), but there's a notable number of false positives (38% FPR). This indicates a need to refine the model to reduce the misclassification of legitimate emails as spam. There is a potential for improvement.

2-2

Again, I chose learning rate as 10^{-4} and performed 1000 iterations. Here are the results for mini-batch gradient ascent and stochastic gradient ascent.

```
Confusion Matrix for Mini-Batch Gradient Ascent:
[[40 25]
 [10 25]]
Performance Metrics for Mini-Batch Gradient Ascent:
Accuracy: 0.65
Precision: 0.62
Recall: 0.80
F1 Score: 0.70
F2 Score: 0.75
Negative Predictive Value (NPV): 0.71
False Positive Rate (FPR): 0.50
False Discovery Rate (FDR): 0.38
```

```
Confusion Matrix for Stochastic Gradient Ascent:
[[49 19]
 [ 1 31]]
Performance Metrics for Stochastic Gradient Ascent:
Accuracy: 0.80
Precision: 0.72
Recall: 0.98
F1 Score: 0.83
F2 Score: 0.91
Negative Predictive Value (NPV): 0.97
False Positive Rate (FPR): 0.38
False Discovery Rate (FDR): 0.28
```

Performance metrics show that stochastic gradient ascent (SGD) outperforms mini-batch gradient ascent for spam classification. SGD achieves higher accuracy, better spam detection (higher recall), and fewer false positives (lower FPR and FDR). This implies that SGD minimizes the misclassification of valid emails while being more successful at accurately identifying spam emails. However, the results are dependent on learning rate, number of iterations and batch size.

Q3 – Support Vector Machines

3-1

In this task, I trained my SVM using the sklearn.svm package. Due to the potentially long computational time with the given C values and ten iterations, I imported the tqdm package to provide visual progress updates during training. I set the validation set size for Monte Carlo Cross Validation (MCCV) to be 0.2 of the total training set. I used MCCV to optimize a linear SVM classifier's C hyperparameter. Using the provided C value, I created a monte_carlo_cv function that shuffles the data repeatedly, divides it into training and validation sets, trains an SVM model, and determines the error rate on the validation set. The result is the average error over a number of iterations. I then experimented with various C values to assess how well they performed, choosing the C value that produced the lowest average error. Then I tested the model with best C value. Average errors for different C values and performance metrics are shown below:

```
100%|██████████| 10/10 [00:02<00:00,  3.53it/s]
Error for C=0.01: 0.1260
100%|██████████| 10/10 [01:04<00:00,  6.42s/it]
Error for C=0.1: 0.0896
100%|██████████| 10/10 [03:25<00:00, 20.51s/it]
Error for C=1: 0.0844
100%|██████████| 10/10 [21:07<00:00, 126.73s/it]
Error for C=10: 0.0823
100%|██████████| 10/10 [59:09<00:00, 354.95s/it]
Error for C=100: 0.1031
100%|██████████| 10/10 [50:23<00:00, 302.40s/it]Error for C=1000: 0.0885
Best C value: 10
```

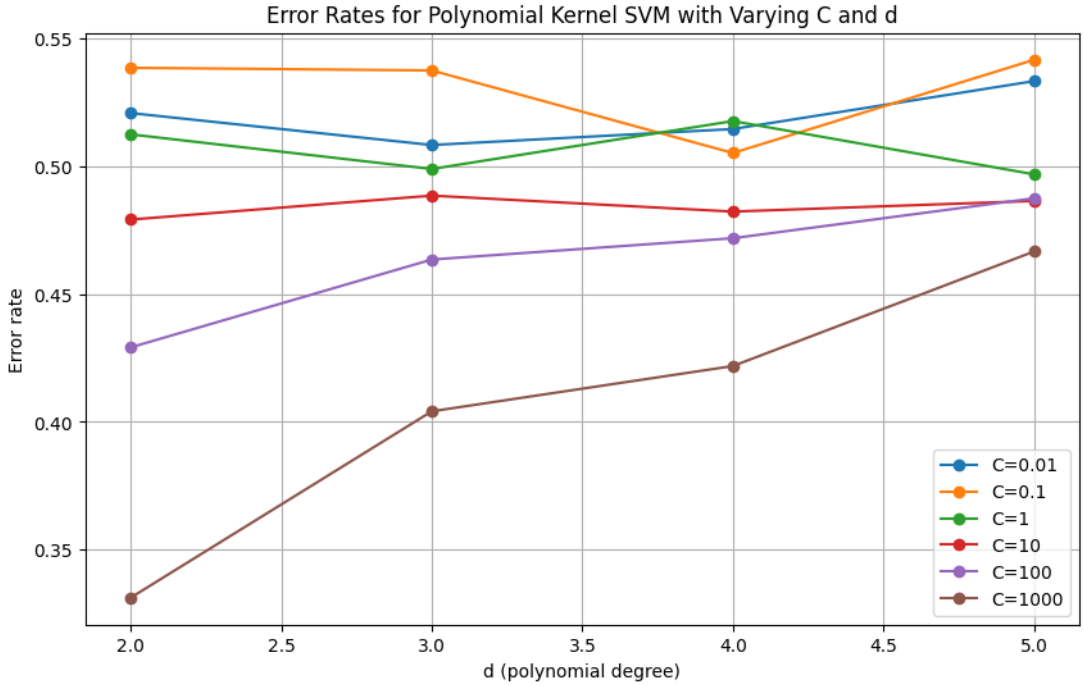
From the output, the best C value is 10. Note that it may change if we run it again since error rates are close to each other and weare randomly selecting data.

```
Confusion Matrix:
 [[47  5]
 [ 3 45]]
Performance Metrics:
Accuracy: 0.92
Precision: 0.90
Recall: 0.94
F1 Score: 0.92
F2 Score: 0.93
Negative Predictive Value (NPV): 0.94
False Positive Rate (FPR): 0.10
False Discovery Rate (FDR): 0.10
```

With C=10, I tested the model with unseen data. Confusion matrix shows that the linear SVM model fitted better than logistic regression models in the previous question. We reached the accuracy value and F1 score of 92%.

3-2

For hyperparameter tuning, I also used Monte Carlo cross-validation, but I modified the SVM model to use a polynomial kernel. As a result, the SVM can potentially become more adept at identifying complex patterns in the data by learning non-linear decision boundaries. The degree 'd' of the polynomial, a new hyperparameter, also needs to be optimized via cross-validation. The basic framework for data shuffle, splitting, training, and evaluation is still the same as it is for the linear SVM implementation. Average errors for different C and d values are shown below:



The results show that we are achieving minimum error at C = 1000 and d = 2. The polynomial SVM's best hyperparameters, as determined by MCCV, were a polynomial degree of two and a C value of 1000. With this configuration, the average error rate was the lowest. The large C indicates that the data may contain non-linear patterns that the SVM is able to adequately model. Due to the relatively low degree, overfitting that could result from using a very high-degree polynomial is avoided. Instead, these non-linearities can be captured with a quadratic transformation. Performance metric for that case is shown below:

```
Confusion Matrix:
 [[25  2]
 [25 48]]
Performance Metrics:
Accuracy: 0.73
Precision: 0.93
Recall: 0.50
F1 Score: 0.65
F2 Score: 0.55
Negative Predictive Value (NPV): 0.66
False Positive Rate (FPR): 0.04
False Discovery Rate (FDR): 0.07
```

If we look at the metrics, we see that polynomial SVM has a worse confusion matrix and performance metrics compared to the linear SVM. This could be due to the fact that the dataset might have a primarily linear structure, where a simpler linear model is sufficient.