DSCI5340.005 Predictive Analytics and Business Forecasting

Homework 4 - Group 1

Md Mohsin Reza (11805929) Safa Kazi (11872240) Umama Khanom Antara (11803668)

Load and Summarize Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 392 entries, 0 to 391
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    int64  
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   year          392 non-null    int64  
 7   origin        392 non-null    int64  
 8   name          392 non-null    object 
dtypes: float64(3), int64(5), object(1)
memory usage: 27.7+ KB
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

1. Creating a Binary Variable

```
mpg_median = 22.75
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	high_mpg
0	18.0	8	307.0	130	3504	12.0	70	1	0
1	15.0	8	350.0	165	3693	11.5	70	1	0
2	18.0	8	318.0	150	3436	11.0	70	1	0
3	16.0	8	304.0	150	3433	12.0	70	1	0
4	17.0	8	302.0	140	3449	10.5	70	1	0

	high_mpg	count
0	0	196
1	1	196

A binary variable 'high_mpg' was created, assigning 1 to cars with MPG above the median and 0 to cars with MPG below the median

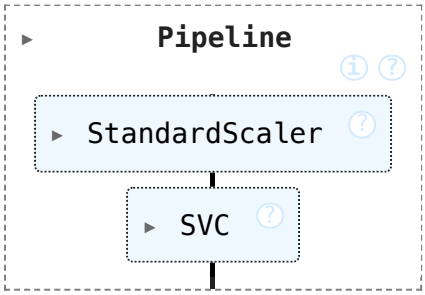
2. Train/test split and linear SVM with different C

Splitting Training Data (80%) and Test Data (20%)

```
(313, 79)
```

Pipeline: StandardScaler + SVC (linear kernel)

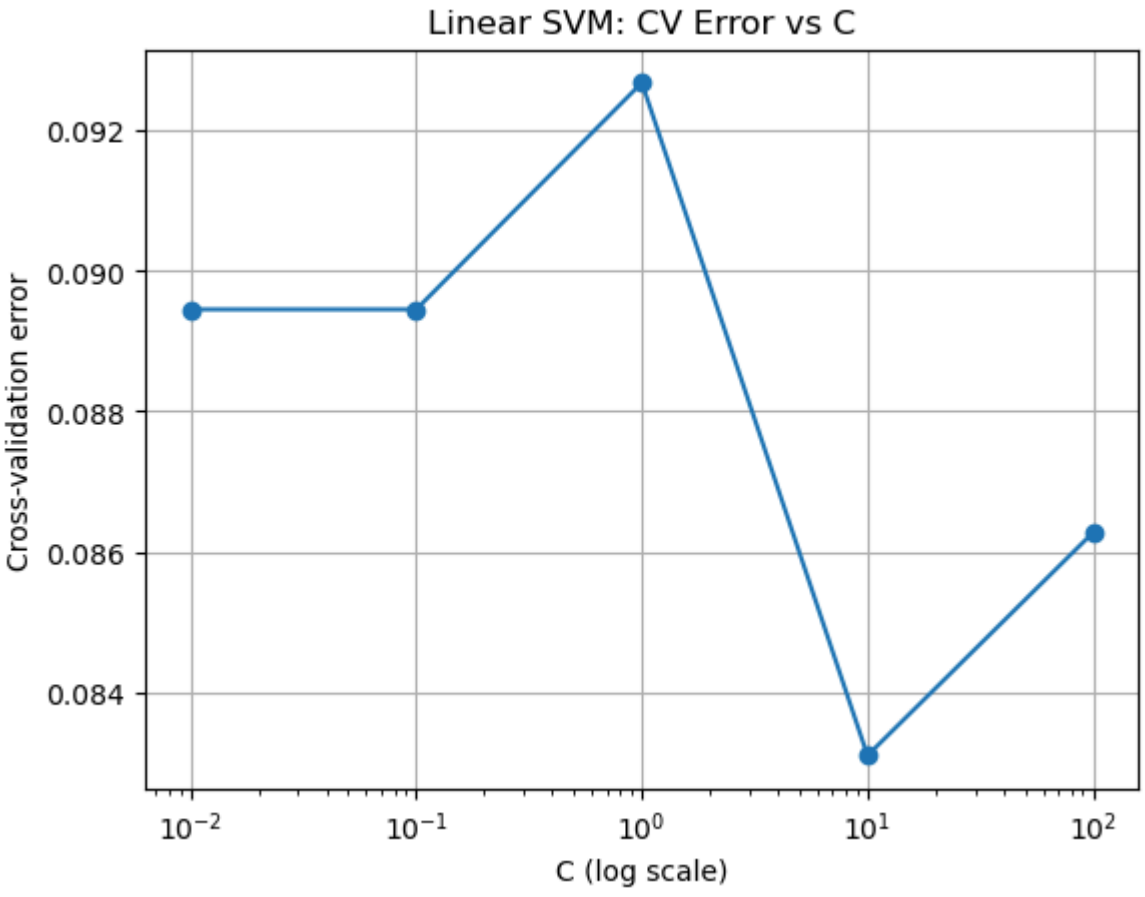
	C	cv_accuracy	cv_error
0	0.01	0.910548	0.089452
1	0.10	0.910548	0.089452
2	1.00	0.907322	0.092678
3	10.00	0.916897	0.083103
4	100.00	0.913722	0.086278



```
Test accuracy: 0.8987341772151899
              precision    recall  f1-score   support

           0       0.97      0.82      0.89        40
           1       0.84      0.97      0.90        39

    accuracy                           0.90        79
   macro avg       0.91      0.90      0.90        79
weighted avg       0.91      0.90      0.90        79

[[33  7]
 [ 1 38]]
```

The results from 5-fold cross-validation showed that C = 10 provided the best performance, with the highest cross-validation accuracy of 0.9169 and the lowest error of 0.0831. Smaller values of C (0.01 and 0.1) gave identical results with an accuracy of 0.9105 and error of 0.0895, indicating similar performance but slightly lower than C = 10. On the other hand, C = 100 resulted in a slightly lower accuracy (0.9137) and higher error (0.0863), suggesting that too much regularization can lead to overfitting. Overall, C = 10 was found to be the optimal value, achieving the best balance between accuracy and error. This value was used to train the final model, which was then evaluated on the test set. The model achieved a test accuracy of 89.87%, demonstrating strong performance in predicting correctly.

3. Cross-validation errors and empirical implications



As C increases, the error initially decreases, reaching its lowest point at C = 10, indicating the best performance with optimal regularization. At very low C values (like 0.01), the model

underfits, resulting in higher cross-validation error. As C increases beyond 10 (e.g., 100), the error rises again, suggesting that the model starts overfitting, fitting the noise in the data. The optimal value for C is therefore 10, as it provides the best balance, ensuring good generalization while avoiding overfitting. This implies that moderate regularization is crucial for achieving the best model performance.

4. SVM with RBF and Polynomial kernels

```
np.float64(1.0)
```

```
np.float64(1.0)
```

```
0.9367088607594937
```

```
np.float64(10.0)
```

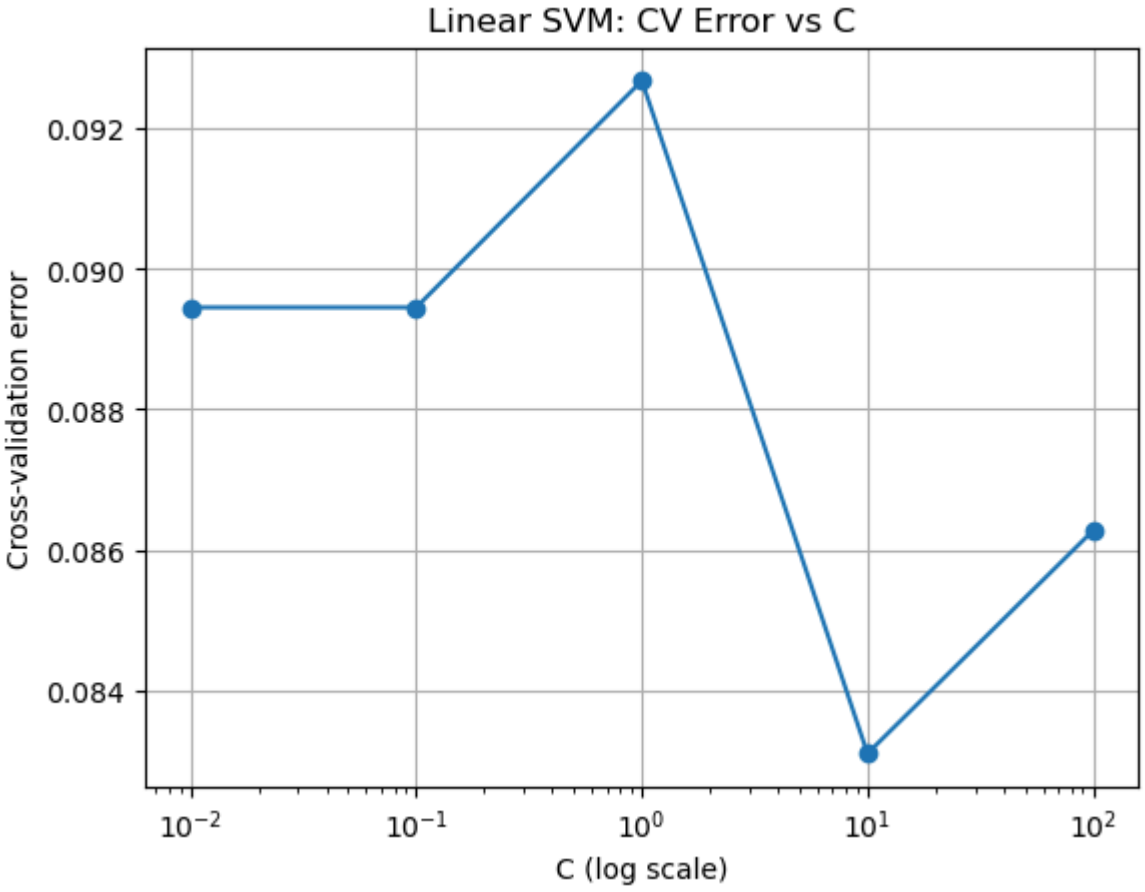
```
np.float64(3.0)
```

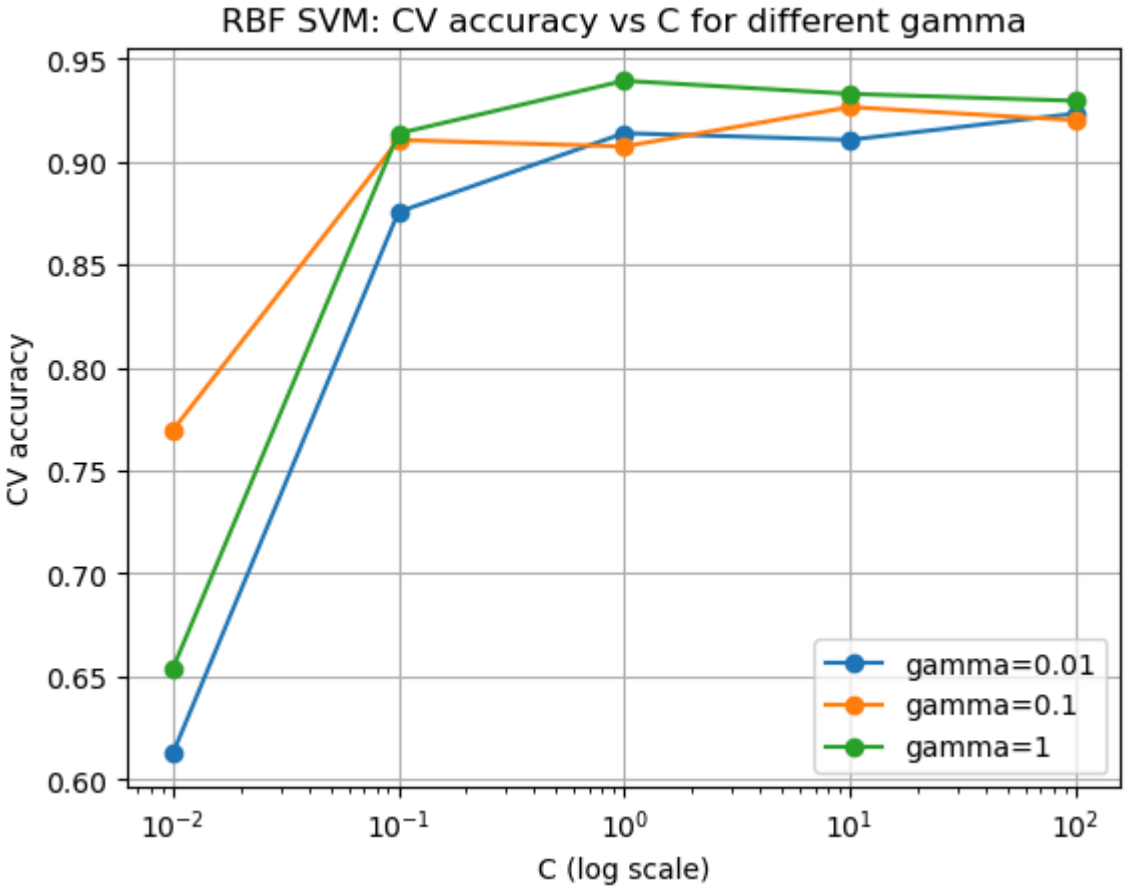
```
0.9113924050632911
```

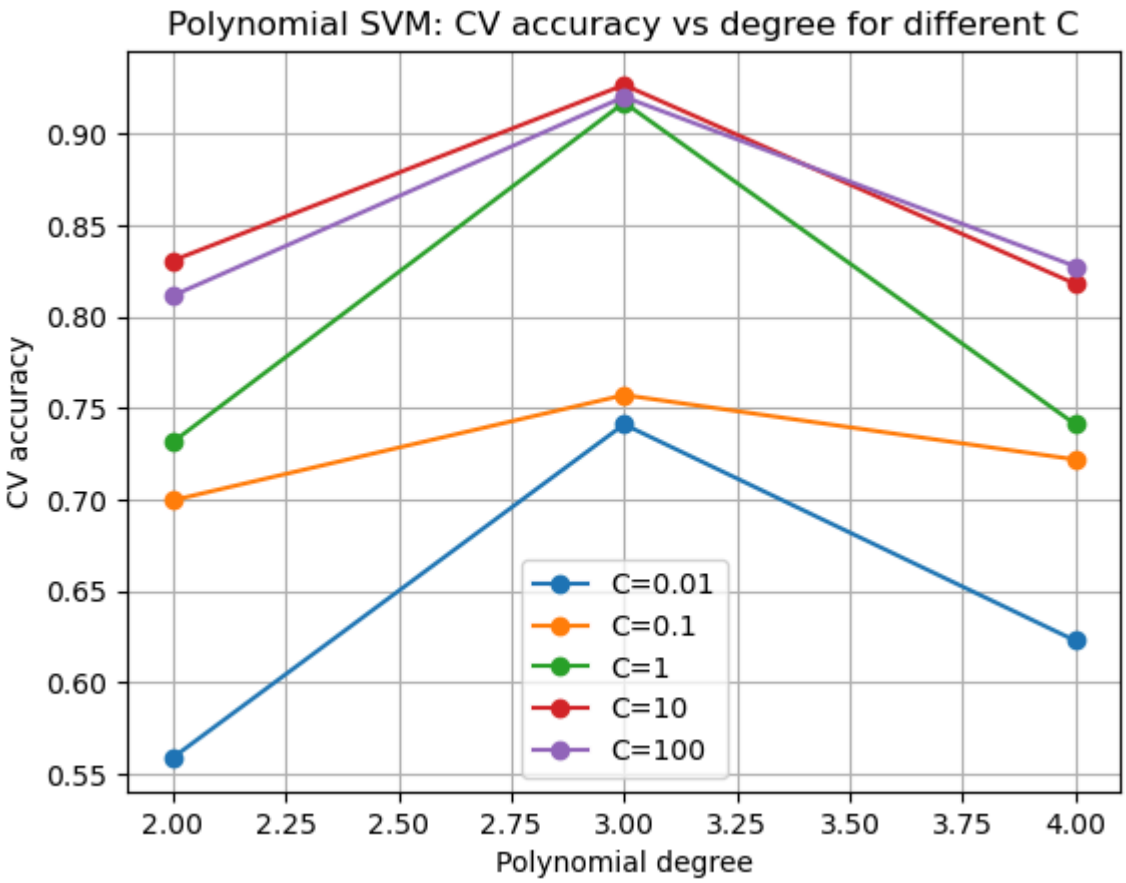
For the RBF SVM, the model was very sensitive to both the C and gamma values. When gamma was small, the model couldn't really capture the pattern in the data, and it kind of underfitted, so the accuracy was lower. But when gamma was too big, the model became too flexible and started fitting almost every small detail, which didn't help on the test set and sometimes made things worse. Also C affects this too-bigger C usually tried harder to classify everything correctly, but sometimes it went too far and overfitted. Hence, the best performance happened when both C and gamma were in a good balanced range, not too small and not too large. This shows the RBF kernel can work really well, but it needs careful tuning or it messes up easily.

When the polynomial SVM was tested with different degrees and C values, the model changed a lot depending on the degree. With a low degree, like 2, the model was too simple and didn't capture enough of the pattern, resulting in a mediocre accuracy. But when the degree got too high, like 4, the model became very complicated and it started overfitting the training data, so the validation accuracy dropped. It also depends on the C value, because a bigger C can make the model try too hard to get every point correct, which sometimes makes things worse. The best results usually happened with a medium degree, like 3, since it gave a good balance between flexibility and not making the model too complex. Overall, polynomial SVM works fine, but it's more sensitive and needs some careful tuning to avoid messing up.

5. Plots to support Q2–Q4


Linear SVM: CV Error vs C
Cross-validation error
0.092
0.090
0.088
0.086
0.084
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
C (log scale)


RBF SVM: CV accuracy vs C for different gamma
CV accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
C (log scale)
gamma=0.01
gamma=0.1
gamma=1


Polynomial SVM: CV accuracy vs degree for different C
CV accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
Polynomial degree
C=0.01
C=0.1
C=1
C=10
C=100