ELL409
Assignment-2
SVM Regression and Classification

Etiksha Jain(2020EE10491)
Sampan Manna(2020EE10547)

Q1

Dataset Description

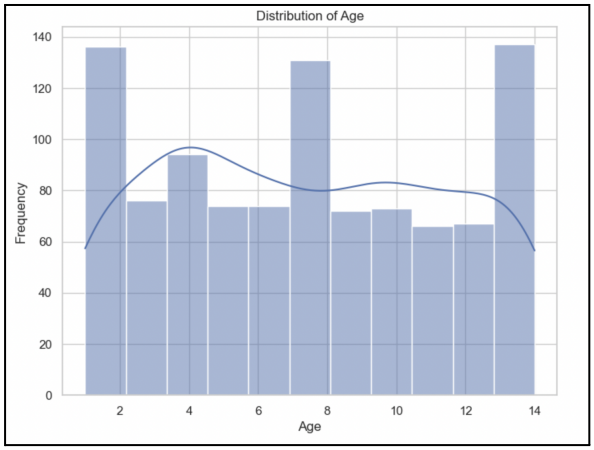
	Age	Mileage	Price
0	13	54785	-1630.906879
1	6	44790	4255.863971
2	1	35735	9848.489578
3	4	10592	5943.521049
4	12	30183	-2310.300853

The given dataset contains a total of 1000 datapoints. Each datapoint has age, mileage and price features.

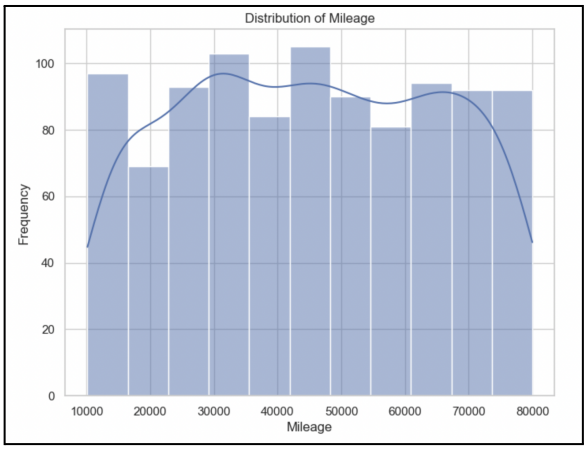
Data Analysis-
Mean Age: 7.357
Max Age: 14
Min Age: 1
Mean Mileage: 45233.642
Max Mileage: 79988
Min Mileage: 10131
Mean Price: 3143.68057702629
Max Price: 16693.499678560598
Min Price: -11073.405596783428

-We split the dataset into train and test using sklearn library
-We normalise the points to have zero mean and unit variance using sklearn library

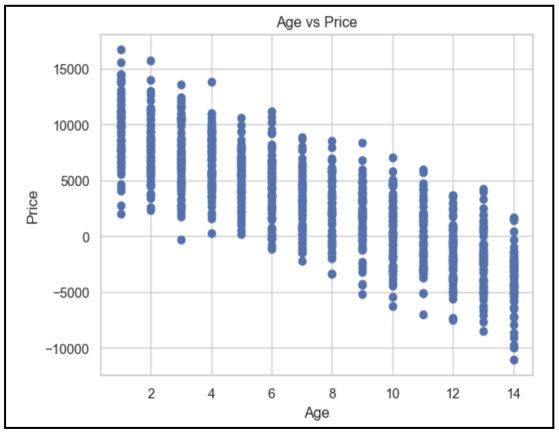
Dataset Visualisation



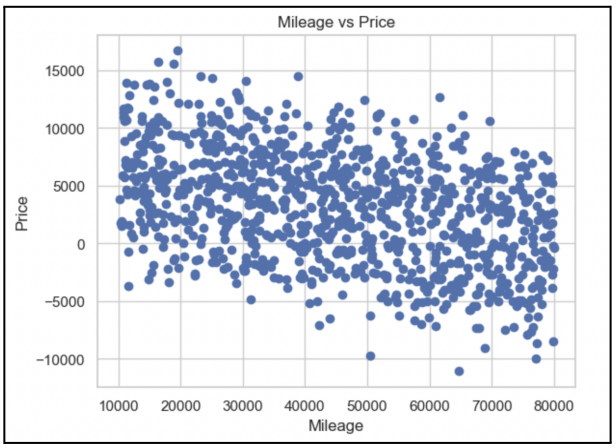
Distribution of Age



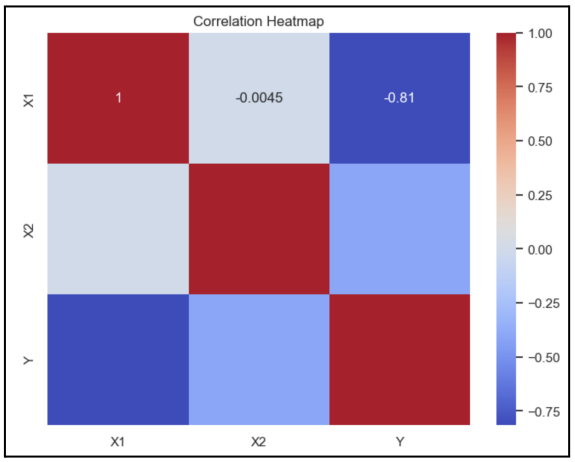
Distribution of Mileage



Dependence of price on Age



Dependence of price on Mileage



Inference

We can see that age and prices are heavily negatively correlated, that is, prices fall as the cars get older. We can find a similar negative correlation between mileage and prices, although it is slightly weaker than the correlation between age and prices.

Model Architecture- SVM Linear Regression Model

As given in the assignment, we used an SVM regression model using linear kernel
The model has been implemented using sklearn library as follows-

```
from sklearn.svm import SVR
svr = SVR(kernel='linear')
svr.fit(X_train, Y_train)
```

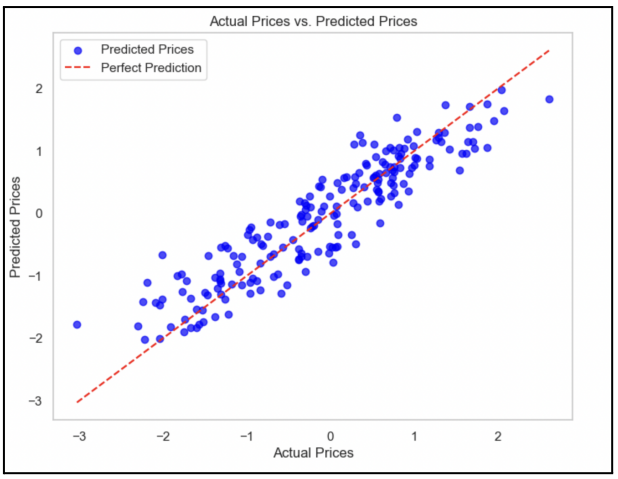
```
SVR(kernel='linear')
```

```
from sklearn.model_selection import GridSearchCV
svr =  svr
param_grid = {'C': [0.1, 1, 10, 100], 'epsilon': [0.1, 0.2, 0.5, 1]}
grid_search = GridSearchCV(svr, param_grid, cv=5)
grid_search.fit(X_train, Y_train)
best_svr = grid_search.best_estimator_
```

We have used **GridSearch** to tune the hyperparameters **C and epsilon**.

Best Model results

- Best model parameters- C=100, epsilon=1
- Mean Absolute Error (MAE): 1658.28
- Mean Squared Error (MSE): 4245854.13
- R-squared (R2): 0.84

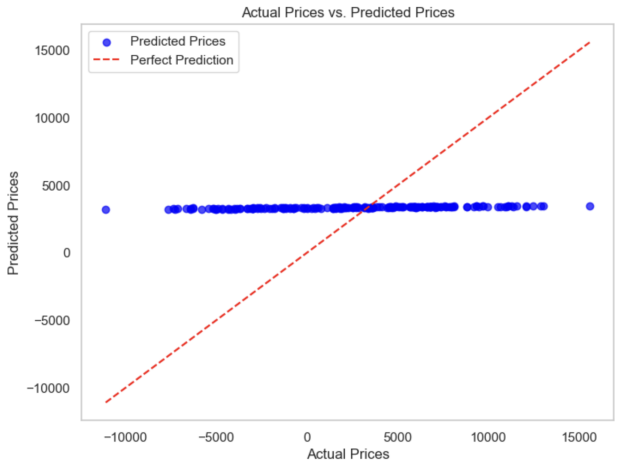


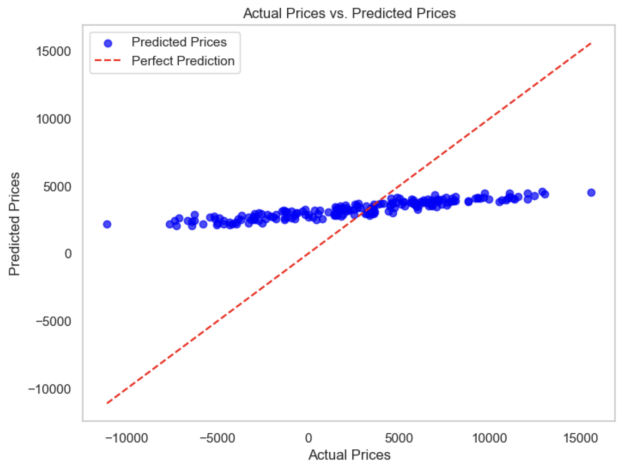
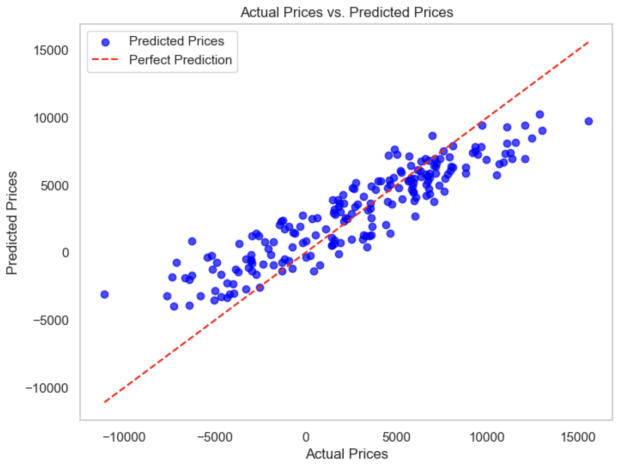
Actual vs Predicted prices

Inference

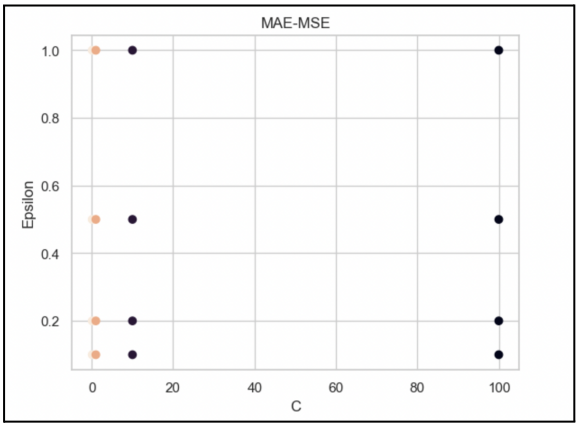
We can see that the results are good with an r2 score of 0.84

Effect of Hyperparameters

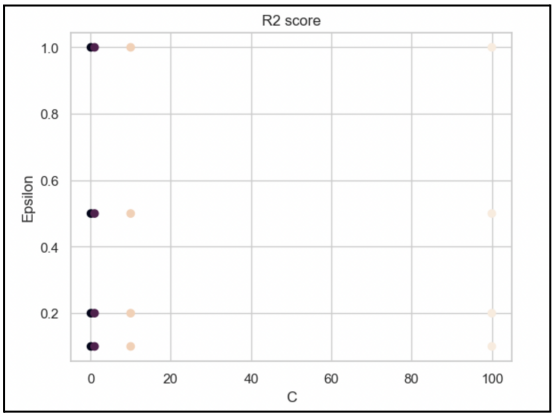
C	Eps	MAE	MSE	R2	Visualisation
0.1	0.1	4210.63	26205322.95	0.01	

1	0.1	3773	21250891.61	0.2	
10	0.1	1951.70	5893355.44	0.78	
100	0.1	1658.29	4245673.37	0.84	

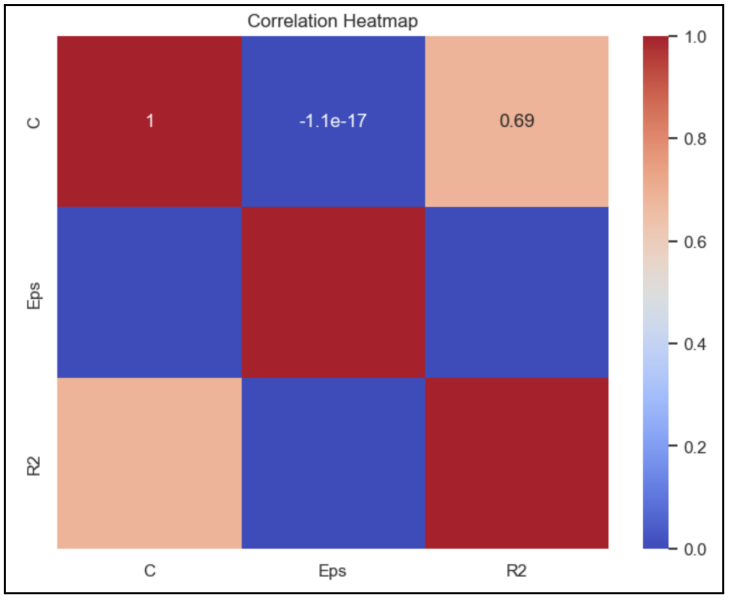
100	1	1658.28	4245854.13	0.84	



Dependence of MAE on C,eps



Dependence of R2 score on C,eps

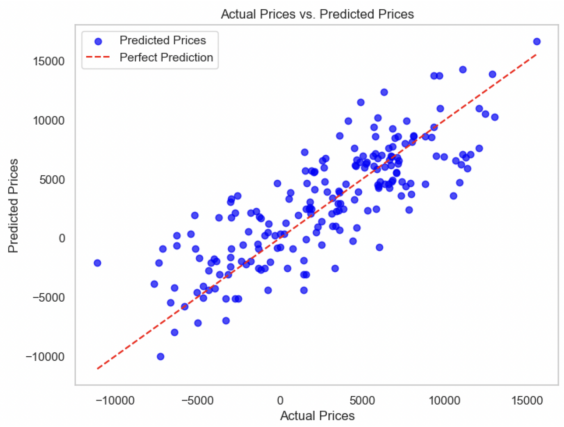
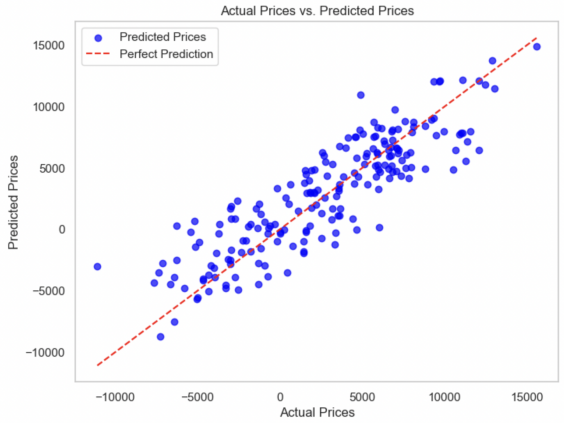


Inference

We can see that R2 and C are positively correlated.

SVM vs Other Regressions

Regression	MAE	MSE	R2	Visualisation
Linear Regression	1638.94	4156483.25	0.84	

Decision Tree Regression	2410.32	9342021.45	0.65	
Random Forest Regression	2002.15	6333297.38	0.76	

Inference

We can see that Linear Regression has the lowest MSE as well as MAE out of the above three cases. The MSE, MAE and R2 values obtained for linear regression are similar to those obtained for the most optimised case of SVM Regression. SVM Regression performs drastically better than Decision Tree Regression and Random Forest Regression.

Conclusion

- The best SVM Regression model obtained gives a R2 of 0.84, Mean Absolute Error (MAE) of 1658.28 and Mean Squared Error (MSE) of 4245854.13
- The linear regression model gives R2 of 0.84, MAE of 1638.94, and MSE of 4156483.25

Q2

CIFAR10 Dataset Description

The CIFAR10 dataset contains a total of 60,000 images. These images are 3x32x32 pixels in size, and each image represents an object. There are 10 classes.All images in the CIFAR10 dataset are colored.

We split the dataset into test and train dataset. The model was trained on trainset and tested on test set.
Total samples = 60,000
Training samples = 50,000
Test samples = 10,000
Image dimensions = 3x32x32
Number of Output class = 10
This is modelled as a Classification problem.

Dataset Preprocessing

-We flatten the images into a vector with 3*32*32 features = 3072 features
-We normalise the feature vectors using sklearn library
-We split the dataset into train and test sets using sklearn library

Dataset Augmentation

We have used ImageGenerator library from keras to generate rotated and flipped images-

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=20,
    horizontal_flip=True,
)

augmented_images = []
augmented_lables = []
for i,image in enumerate(cifar_trainset.data):
    augmented_images.append(image/255.0)
    # Apply data augmentation
    augmented_images.append(datagen.random_transform(image/255.0))
    augmented_lables.append(cifar_trainset.targets[i])
    augmented_lables.append(cifar_trainset.targets[i])
augmented_images = np.array(augmented_images)
```

The new dataset size after data augmentation=**(100000,32,32,3)**

Feature Extraction

Curse of Dimensionality: In high-dimensional datasets, the number of features can be much larger than the number of observations, leading to increased computational complexity and

potential overfitting. PCA helps address the curse of dimensionality by reducing the number of features while preserving the most important information.

We apply PCA to reduce the feature size from 3072 to 500 using sklearn library-

```
# Perform PCA for dimensionality reduction
from sklearn.decomposition import PCA
def perform_pca(features, n_components=500):
    pca = PCA(n_components=n_components)
    return pca.fit_transform(features)
```

SVM Linear Classification Model

As given in the assignment, we used an SVM regression model using linear kernel
The model has been implemented using sklearn library as follows-

```
def svm_linear(cc):
    svc = svm.SVC(kernel = 'linear', C = cc)
    svc.fit(train_data_pca, train_labels)
    return svc
```

Results-

Model-https://drive.google.com/file/d/1lCy5MHY2Xbk3Fgy3DY3ud94nfoUSKF0J/view?usp=drive_link

We got a maximum confidence **score=40%** using **GridSearch** from sklearn library. The training results are as follows-

```
Fitting 2 folds for each of 3 candidates, totalling 6 fits
[CV 1/2] END ..............................C=100;, score=0.254 total time=   0.1s
[CV 2/2] END ..............................C=10;, score=0.258 total time=   0.1s
[CV 2/2] END ................................C=1;, score=0.344 total time=  25.6s
[CV 1/2] END ................................C=1;, score=0.351 total time=  25.4s
[CV 1/2] END ................................C=1;, score=0.254 total time=   0.1s
[CV 1/2] END ..............................C=10;, score=0.354 total time= 4.0min
[CV 1/2] END ..............................C=10;, score=0.254 total time=   0.1s
[CV 2/2] END ..............................C=10;, score=0.344 total time= 4.2min
[CV 2/2] END .............................C=100;, score=0.258 total time=   0.1s
[CV 1/2] END .............................C=100;, score=0.355 total time=31.2min
[CV 2/2] END ................................C=1;, score=0.254 total time=   0.1s
[CV 2/2] END .............................C=100;, score=0.344 total time=34.1min
```

Best Model-

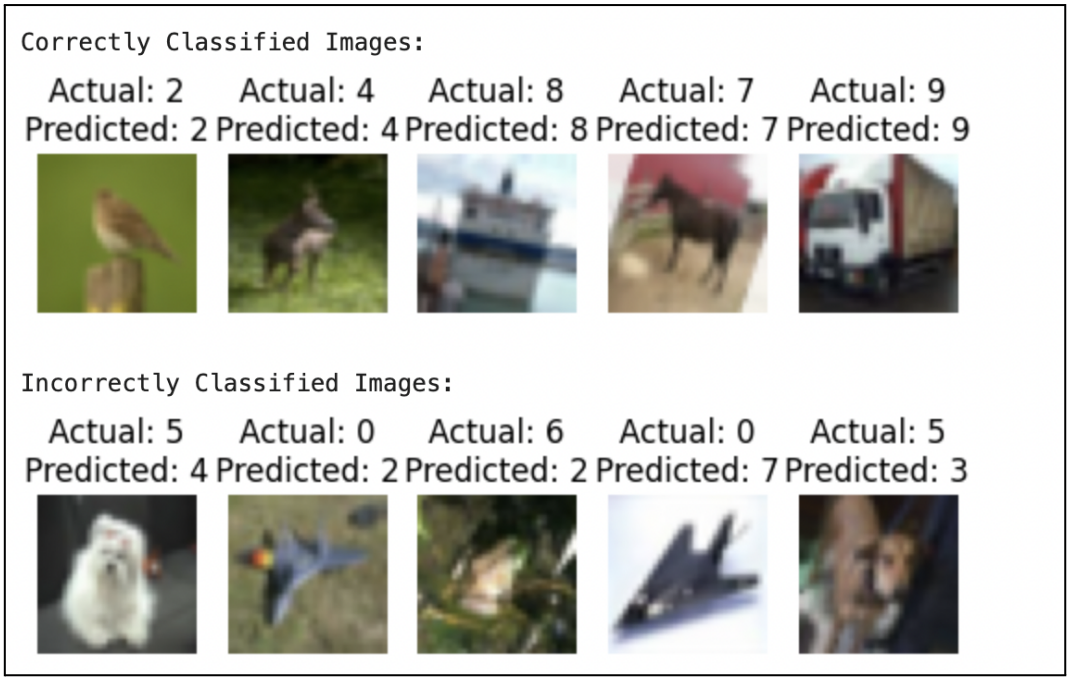
- C=100

Results metrics for best model-

- Train accuracy: 0.386625
- Test accuracy: 0.178

- Precision: 0.17804723291236985
- Recall: 0.178
- F1 score: 0.1774955941846392

Correctly and incorrectly classified images-



SVM RBF Classification Model

As given in the assignment, we used an SVM regression model using linear kernel
The model has been implemented using sklearn library as follows-

```
def svm_rbf(cc,g):
    svc = svm.SVC(kernel = 'rbf', C = cc, gamma = g)
    svc.fit(train_data_pca, train_labels)
    return svc
```

Results-

Model-https://drive.google.com/file/d/19_g1iLPXPunD22qY5IbLWUnomc-6uZfK/view?usp=drive_link

We got a maximum confidence **score=57%** using **GridSearch** from sklearn library. The training results are as follows-

```
Fitting 3 folds for each of 6 candidates, totalling 18 fits
[CV 1/3] END .................C=1, gamma=0.001;, score=0.455 total time=200.2min
[CV 3/3] END ...............C=10, gamma=0.0001;, score=0.423 total time=111.6min
[CV 3/3] END .................C=1, gamma=0.001;, score=0.456 total time=195.2min
[CV 2/3] END ...............C=10, gamma=0.0001;, score=0.429 total time=116.8min
[CV 2/3] END .................C=1, gamma=0.001;, score=0.465 total time=200.2min
[CV 1/3] END ...............C=100, gamma=0.001;, score=0.565 total time=112.9min
[CV 1/3] END ................C=1, gamma=0.0001;, score=0.381 total time=236.1min
[CV 3/3] END ................C=100, gamma=0.001;, score=0.564 total time=85.0min
[CV 3/3] END ................C=1, gamma=0.0001;, score=0.383 total time=236.0min
[CV 2/3] END ................C=100, gamma=0.001;, score=0.572 total time=85.0min
[CV 2/3] END ................C=1, gamma=0.0001;, score=0.389 total time=236.1min
[CV 1/3] END ...............C=100, gamma=0.0001;, score=0.459 total time=81.3min
[CV 2/3] END ................C=10, gamma=0.001;, score=0.523 total time=179.4min
[CV 1/3] END ...............C=10, gamma=0.0001;, score=0.421 total time=127.8min
[CV 3/3] END ...............C=100, gamma=0.0001;, score=0.459 total time=57.6min
[CV 1/3] END ................C=10, gamma=0.001;, score=0.516 total time=179.2min
[CV 3/3] END ................C=10, gamma=0.001;, score=0.512 total time=121.5min
[CV 2/3] END ...............C=100, gamma=0.0001;, score=0.469 total time=49.4min
```

Best Model-

- C=100
- gamma=0.001

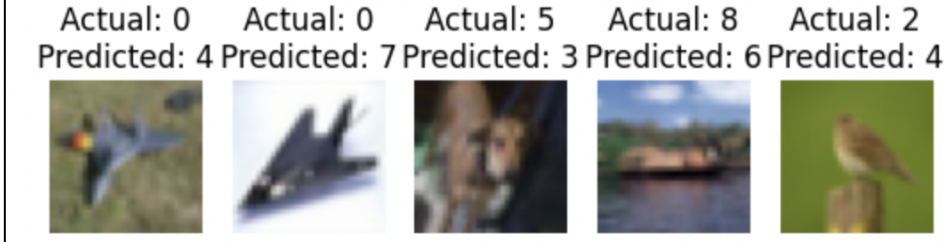
Results metrics for best model-

- Train accuracy: 0.7555375
- Test accuracy: 0.2002
- Precision: 0.19960996273701967
- Recall: 0.2002
- F1 score: 0.19910753738458453

Correctly and incorrectly classified images-

Correctly Classified Images:

Incorrectly Classified Images:



Conclusion

- The maximum maximum confidence score using GridSearch was 40% for the linear kernel and 57% for the RBF kernel
- We get a higher test set accuracy (0.2002) for the RBF Classification implementation than the one using the linear SVM Classifier (0.178)
- The best model for the RBF Classification had the hyperparameters C=100 and γ=0.001, and the one using the linear SVM Classifier had hyperparameter C=100.

Q3.

Stock Data

Date	Open	High	Low	Close	Adj Close	Volume
2023-10-16	176.750000	179.080002	176.509995	178.720001	178.484863	52517000
2023-10-17	176.649994	178.419998	174.800003	177.149994	176.916916	57549400
2023-10-18	175.580002	177.580002	175.110001	175.839996	175.608643	54764400
2023-10-19	176.039993	177.839996	175.190002	175.460007	175.229156	59302900
2023-10-20	175.309998	175.419998	172.639999	172.880005	172.652542	64189300

Stock Features and Labels

We use percentage change between high and low stock price and opening and closing stock price as input features rather than original prices.
To predict the output for the next 30 days, we take the Opening and Adjusted closing price of the 30th day after that day as the output for that datapoint-

```
stock_data = stock_data[['Open',  'High',  'Low',  'Adj Close', 'Volume']]
stock_data['HL_PCT'] = (stock_data['High'] - stock_data['Low']) / stock_data['Adj Close'] * 100.0
stock_data['PCT_change'] = (stock_data['Adj Close'] - stock_data['Open']) / stock_data['Open'] * 100.0

stock_data = stock_data[['Open','Adj Close', 'HL_PCT', 'PCT_change', 'Volume']]
print(stock_data.head())
```

Using percentage changes as features

```
forecast_out = 30

stock_data['label_adj_close'] = stock_data['Adj Close'].shift(-forecast_out)
stock_data['label_open'] = stock_data['Open'].shift(-forecast_out)
# df=df.dropna(inplace=True)
```

Using the value of 30th day after that day as label

Final Stock Data

Date	Open	Adj Close	HL_PCT	PCT_change	Volume	label_adj_close	label_open
2023-10-16	176.750000	178.484863	1.439902	0.981535	52517000	172.772385	170.910004
2023-10-17	176.649994	176.916916	2.046155	0.151102	57549400	173.211807	173.050003
2023-10-18	175.580002	175.608643	1.406537	0.016312	54764400	170.874893	171.880005
2023-10-19	176.039993	175.229156	1.512302	-0.460598	59302900	166.670425	170.369995
2023-10-20	175.309998	172.652542	1.610170	-1.515861	64189300	167.998672	166.910004
2023-10-23	170.910004	172.772385	2.361490	1.089685	55980100	170.065933	169.020004

News Data

	title	description	pubAt
0	Apple TV+ prices have doubled in just over a year	Apple is jacking up the prices of several of i...	2023-10-25
1	Apple TV Plus is getting a price hike — and ot...	Apple is increasing the price on Apple Arcade,...	2023-10-25
2	All the New Ways to Use Your Apple Watch	With watchOS 10, Apple breathed new life into ...	2023-11-02
3	Apple HomeKit Secure Video: Pros and Cons	Shopping for a security camera or video doorbe...	2023-11-08
4	How to connect a Nest Thermostat to Apple Home...	The Nest Thermostat now works with Apple Home ...	2023-10-30

Merged Dataset

We merge the news and stock datasets according to their respective dates-

	Open	Adj Close	HL_PCT	PCT_change	Volume	label_adj_close	label_open	title	description	pubAt
0	176.750000	178.484863	1.439902	0.981535	52517000	172.772385	170.910004	Goldman Sachs might be trying to offload Apple...	Goldman Sachs, Apple's banking partner for its...	2023-10-16
1	176.750000	178.484863	1.439902	0.981535	52517000	172.772385	170.910004	Goldman Sachs Wants Out of the Apple Card Busi...	Goldman Sachs' third-quarter earnings may be t...	2023-10-16
2	176.750000	178.484863	1.439902	0.981535	52517000	172.772385	170.910004	Apple's Reportedly Making Progress on a Cheape...	Apple is making progress on a cheaper version ...	2023-10-16
3	176.750000	178.484863	1.439902	0.981535	52517000	172.772385	170.910004	Hands-on: Meta Quest 3 is the Apple Vision Pro...	I've tried them both, and the new Meta Quest 3...	2023-10-16
4	176.750000	178.484863	1.439902	0.981535	52517000	172.772385	170.910004	iPhone Loses Market Share Dominance in China, ...	Apple's iPhone lost its top spot among Chinese...	2023-10-16

Sentiment Score

For every news description, we want to use the corresponding sentiment as a feature in predicting the stock price. Therefore, we want to give a sentiment score to each news description (and not titles since they can be deceptive):

```
sentiment_labels = ["Very Negative", "Negative", "Neutral", "Positive", "Very Positive"]
```

We use the keras **pre-trained BertClassifier** to classify the news as positive (will increase stock price) or negative (will decrease stock price)

SVM Regression RBF Model - From Scratch

```
class SVR_RBF_Kernel:
    def __init__(self, C=1, epsilon=0.1, gamma=0.1):
        self.C = C  # regularization parameter
        self.epsilon = epsilon  # epsilon-tube for insensitive loss function
        self.gamma = gamma  # RBF kernel parameter
        self.support_vectors = None
        self.support_vector_labels = None
        self.support_vector_alphas = None
        self.b = None
```

```
    def fit(self, X, y):
        n_samples, n_features = X.shape

        # Gram matrix
        K = self.rbf_kernel(X, X)

        # Convert to cvxopt matrix format
        P = cvxopt.matrix(np.outer(y, y) * K)
        q = cvxopt.matrix(-1 * np.ones(n_samples))
        G = cvxopt.matrix(np.vstack((-np.eye(n_samples), np.eye(n_samples))))
        h = cvxopt.matrix(np.hstack((np.zeros(n_samples), np.ones(n_samples) * self.C)))

        A = cvxopt.matrix(y, (1, n_samples))
        b = cvxopt.matrix(0.0)

        # Solve the quadratic programming problem
        solution = cvxopt.solvers.qp(P, q, G, h, A, b)

        # Lagrange multipliers (alphas)
        alphas = np.ravel(solution['x'])
        print('alphas')
        print(alphas)

        # Support vectors have non-zero lagrange multipliers
        sv = alphas > 1e-13
        self.support_vectors = X[sv]
        self.support_vector_labels = y[sv]
        self.support_vector_alphas = alphas[sv]

        # Intercept term
        self.b = np.mean(self.support_vector_labels - np.sum(self.support_vector_alphas * self.support_vector_labels * self.rbf_kernel(self.
```

Results- Apple

Adjusted Closing Price

- Predicted price=171.517
- Mean Absolute Error (MAE): 4.57
- Mean Squared Error (MSE): 42.61
- R-squared (R2): -0.34
- Confidence: 42.61

Opening Price

- Predicted price=171.517
- Mean Absolute Error (MAE): 3.66
- Mean Squared Error (MSE): 33.52
- R-squared (R2): -0.23
- Confidence: 33.52

Results- Google

Adjusted Closing Price

- Predicted price=126.5123257
- Mean Absolute Error (MAE): 6.09
- Mean Squared Error (MSE): 58.17
- R-squared (R2): -0.84
- Confidence: 58.17

Opening Price

- Predicted price=125.633
- Mean Absolute Error (MAE): 6.09
- Mean Squared Error (MSE): 58.17
- R-squared (R2): -0.84
- Confidence: 58.17

Results- Meta

Adjusted Closing Price

- Predicted price=297.9723257
- Mean Absolute Error (MAE): 10.34
- Mean Squared Error (MSE): 173.01
- R-squared (R2): -0.81
- Confidence: 173.01

Opening Price

- Predicted price=300.455
- Mean Absolute Error (MAE): 10.57
- Mean Squared Error (MSE): 147.76
- R-squared (R2): -1.07
- Confidence: 147.76

Conclusion

- In this part of the assignment, we merged the news dataset and the stock dataset and used the keras pre-trained BertClassifier to create a new feature which is news sentiment, which is positive or negative depending upon how it affects the stocks

- We ran the model on three companies, Apple, Google and Meta. The MAE and MSE obtained were maximum for Meta, followed by Google, and the least for Apple.
- Notably, the R2 value came out negative, indicating that the performance of the model was worse than that of a horizontal straight line. The model fails to capture the pattern in the dataset. This is because the available dataset was very limited for both training and testing.