Q3. **COVID - Drug Discovery for COVID19**

In order to learn the vector representations of molecular substructures, we use the mol2vec library. Just like word2vec, which learns embeddings such that vectors representing similar words group together, Mol2vec is also an unsupervised machine learning approach, used to obtain high dimensional embeddings of chemical substructures.

We are using a pre-trained model ‘model_300dim.pkl’

Packages/lbraries needed to be installed :

- Anaconda
- rdkit
- mol2vec

Input format :

We use command line arguments to take the input of the paths of the files.
We take the path of the training data csv file as our first command line argument and path of test data file as our second argument.
The path of the pre-trained model is stored in the local variable ‘model_path’.

Data preprocessing :

After loading the dataset, we proceed as follows :

1. We transform SMILES representations to molecular representations.

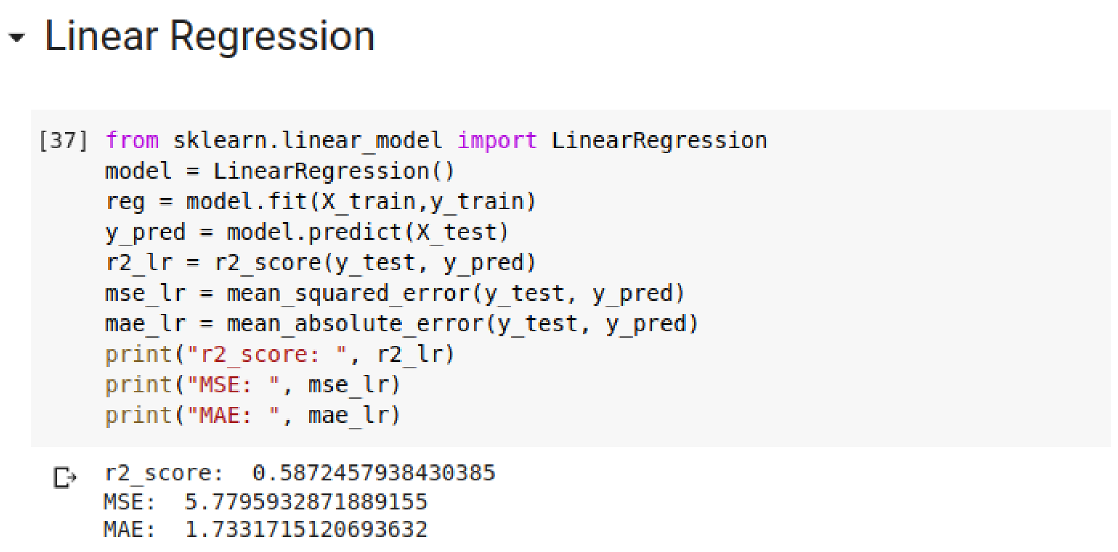
2. We load the pre-trained mol2vec model.

3. We import functions to construct the molecular sentence and generate molecular embeddings using the trianed model.

Model selection :

After performing train validation split on the dataset, we test the r2 score, MSE and MAE of the different models. The outcomes are shown below -

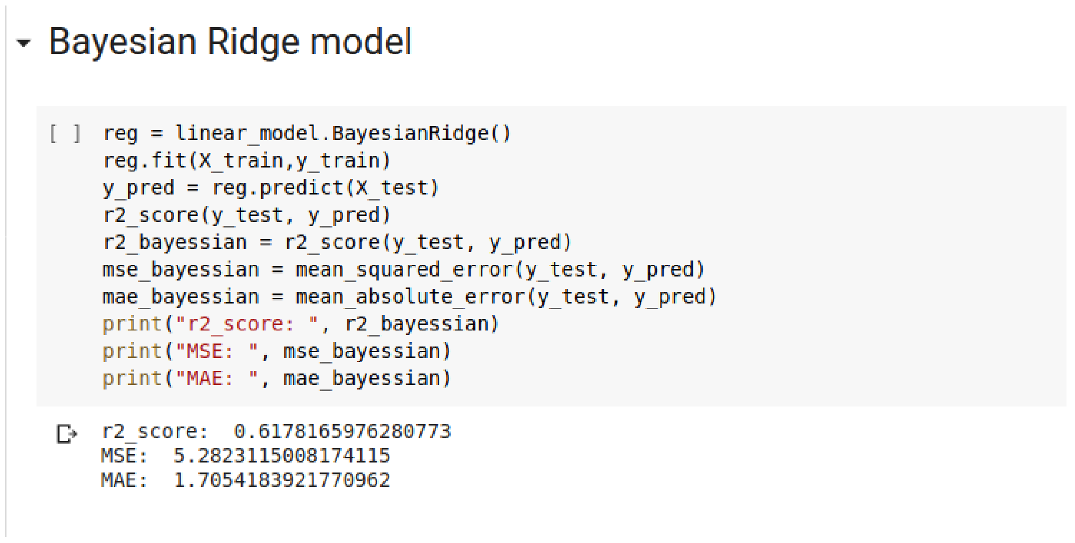
Outcomes:



Linear Regression

```
[37] from sklearn.linear_model import LinearRegression
     model = LinearRegression()
     reg = model.fit(X_train,y_train)
     y_pred = model.predict(X_test)
     r2_lr = r2_score(y_test, y_pred)
     mse_lr = mean_squared_error(y_test, y_pred)
     mae_lr = mean_absolute_error(y_test, y_pred)
     print("r2_score: ", r2_lr)
     print("MSE: ", mse_lr)
     print("MAE: ", mae_lr)
```

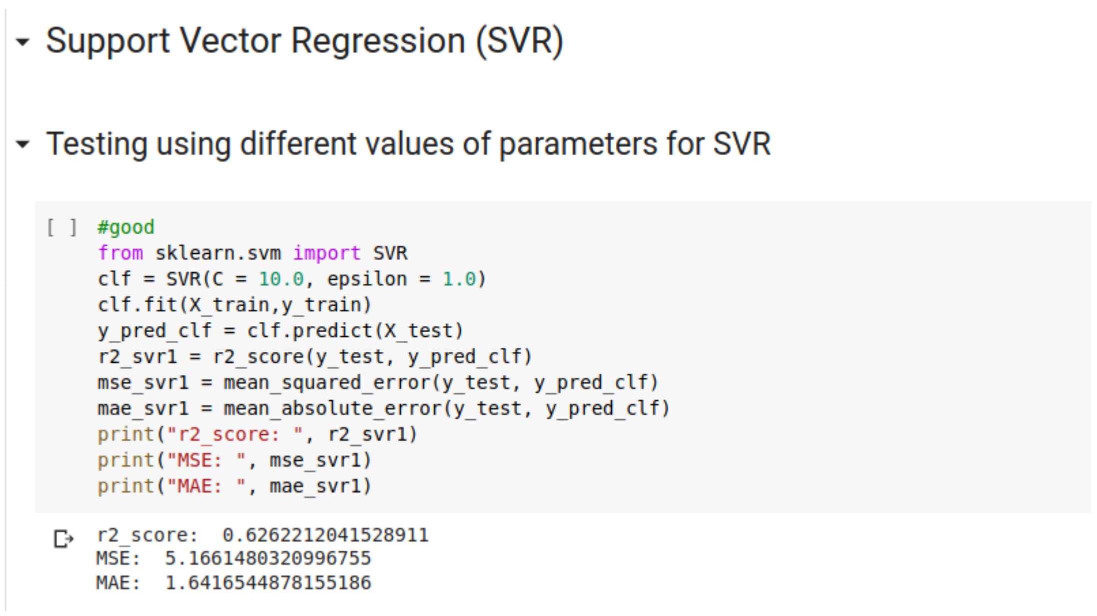
```
r2_score:  0.5872457938430385
MSE:  5.7795932871889155
MAE:  1.7331715120693632
```



Bayesian Ridge model

```
[ ] reg = linear_model.BayesianRidge()
    reg.fit(X_train,y_train)
    y_pred = reg.predict(X_test)
    r2_score(y_test, y_pred)
    r2_bayessian = r2_score(y_test, y_pred)
    mse_bayessian = mean_squared_error(y_test, y_pred)
    mae_bayessian = mean_absolute_error(y_test, y_pred)
    print("r2_score: ", r2_bayessian)
    print("MSE: ", mse_bayessian)
    print("MAE: ", mae_bayessian)
```

```
r2_score:  0.6178165976280773
MSE:  5.2823115008174115
MAE:  1.7054183921770962
```



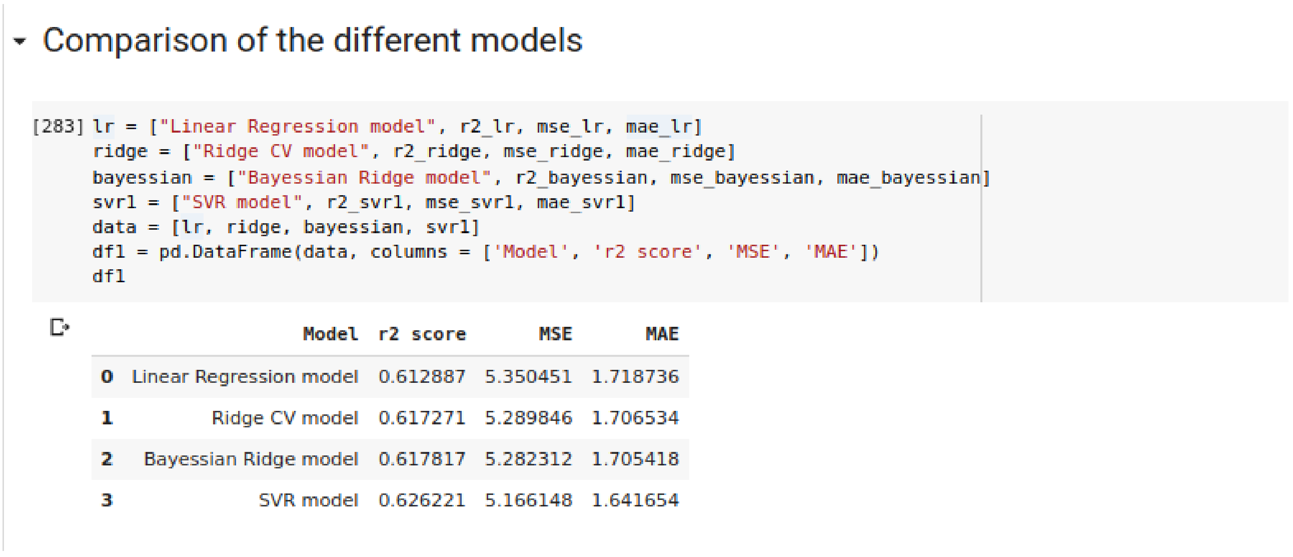
Support Vector Regression (SVR)

Testing using different values of parameters for SVR

```
[ ] #good
    from sklearn.svm import SVR
    clf = SVR(C = 10.0, epsilon = 1.0)
    clf.fit(X_train,y_train)
    y_pred_clf = clf.predict(X_test)
    r2_svr1 = r2_score(y_test, y_pred_clf)
    mse_svr1 = mean_squared_error(y_test, y_pred_clf)
    mae_svr1 = mean_absolute_error(y_test, y_pred_clf)
    print("r2_score: ", r2_svr1)
    print("MSE: ", mse_svr1)
    print("MAE: ", mae_svr1)
```

```
r2_score:  0.6262212041528911
MSE:  5.1661480320996755
MAE:  1.6416544878155186
```

Comparison :

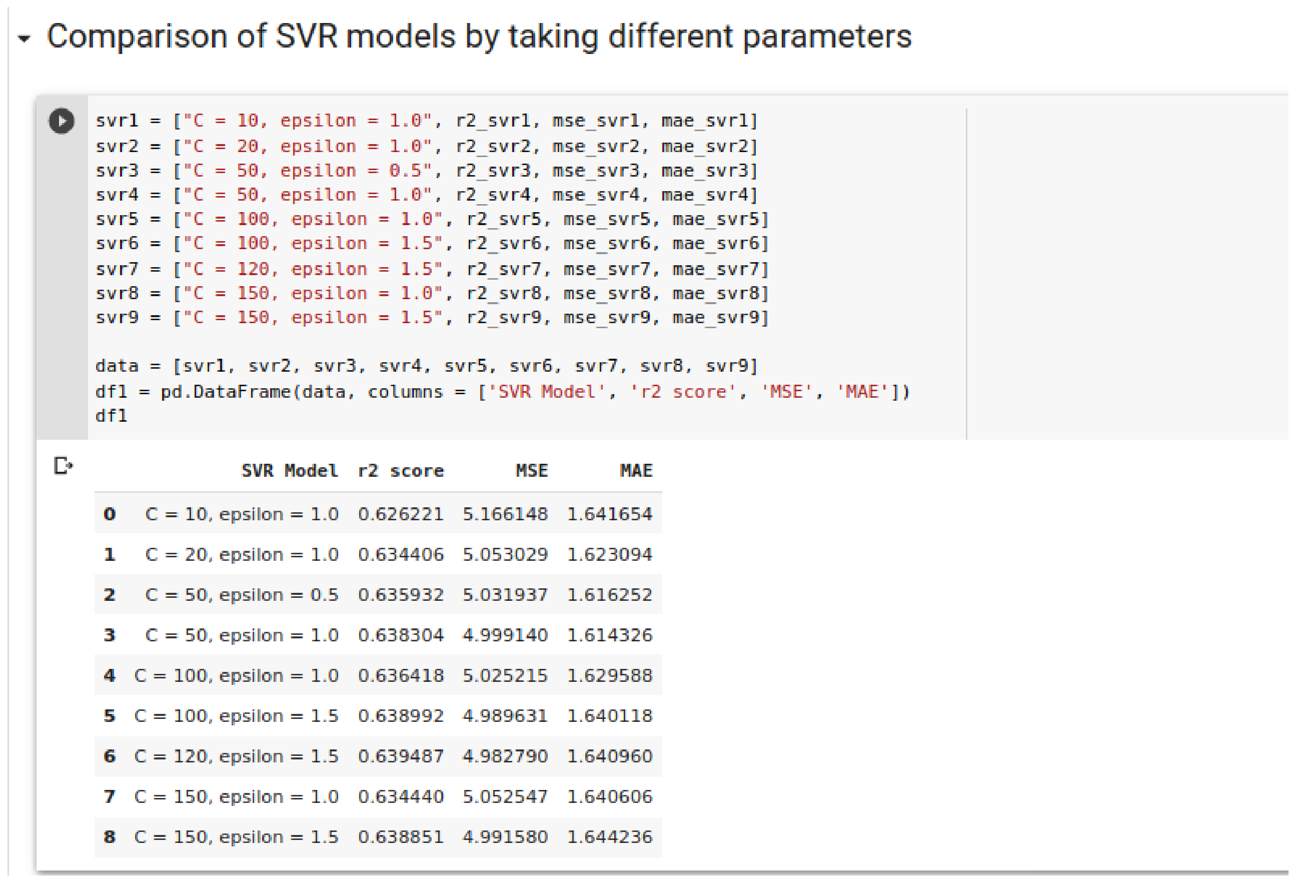


Comparison of the different models

```
[283] lr = ["Linear Regression model", r2_lr, mse_lr, mae_lr]
      ridge = ["Ridge CV model", r2_ridge, mse_ridge, mae_ridge]
      bayessian = ["Bayessian Ridge model", r2_bayessian, mse_bayessian, mae_bayessian]
      svr1 = ["SVR model", r2_svr1, mse_svr1, mae_svr1]
      data = [lr, ridge, bayessian, svr1]
      df1 = pd.DataFrame(data, columns = ['Model', 'r2 score', 'MSE', 'MAE'])
      df1
```

	Model	r2 score	MSE	MAE
0	Linear Regression model	0.612887	5.350451	1.718736
1	Ridge CV model	0.617271	5.289846	1.706534
2	Bayessian Ridge model	0.617817	5.282312	1.705418
3	SVR model	0.626221	5.166148	1.641654

From the above table showing the performance of different models, we see that Support Vector Regression (SVR) performed the best. So, we choose SVR as our final model and test using different values of parameters for SVR.



Comparison of SVR models by taking different parameters

```
svr1 = ["C = 10, epsilon = 1.0", r2_svr1, mse_svr1, mae_svr1]
svr2 = ["C = 20, epsilon = 1.0", r2_svr2, mse_svr2, mae_svr2]
svr3 = ["C = 50, epsilon = 0.5", r2_svr3, mse_svr3, mae_svr3]
svr4 = ["C = 50, epsilon = 1.0", r2_svr4, mse_svr4, mae_svr4]
svr5 = ["C = 100, epsilon = 1.0", r2_svr5, mse_svr5, mae_svr5]
svr6 = ["C = 100, epsilon = 1.5", r2_svr6, mse_svr6, mae_svr6]
svr7 = ["C = 120, epsilon = 1.5", r2_svr7, mse_svr7, mae_svr7]
svr8 = ["C = 150, epsilon = 1.0", r2_svr8, mse_svr8, mae_svr8]
svr9 = ["C = 150, epsilon = 1.5", r2_svr9, mse_svr9, mae_svr9]

data = [svr1, svr2, svr3, svr4, svr5, svr6, svr7, svr8, svr9]
df1 = pd.DataFrame(data, columns = ['SVR Model', 'r2 score', 'MSE', 'MAE'])
df1
```

	SVR Model	r2 score	MSE	MAE
0	C = 10, epsilon = 1.0	0.626221	5.166148	1.641654
1	C = 20, epsilon = 1.0	0.634406	5.053029	1.623094
2	C = 50, epsilon = 0.5	0.635932	5.031937	1.616252
3	C = 50, epsilon = 1.0	0.638304	4.999140	1.614326
4	C = 100, epsilon = 1.0	0.636418	5.025215	1.629588
5	C = 100, epsilon = 1.5	0.638992	4.989631	1.640118
6	C = 120, epsilon = 1.5	0.639487	4.982790	1.640960
7	C = 150, epsilon = 1.0	0.634440	5.052547	1.640606
8	C = 150, epsilon = 1.5	0.638851	4.991580	1.644236

What does the parameters mean ?
Epsilon : In the epsilon-SVR model, the value of epsilon defines a margin of tolerance where no penalty is given to errors.

C denotes the regularization parameter.
In SVM, the optimization problem is to optimize the fit of the line to data as well as penalize the amount of samples inside the margin. *C* denotes the weight of how much samples inside the margin contribute to the overall error. Thus, with a low C, samples inside the margins are penalized less, and with a high value of C, they are penalized more.
In other words, we can think of it this way :
Large Value of parameter C means small margin is allowed

Small Value of paramerter C means arge margin is allowed

Thus we test with different values of these parameters and find that a high value of C gives us better results. From the above observations, we choose a C value close to 100 and epsilon value of 1 in our final model.

We get RMSE around 2.22 on the final test data.