Predicting Vehicle CO_2 Emissions Using Support Vector Regression (SVR)

Author: Md Athar Shihab Biplob
Student ID: 24082155
Module: Machine Learning and Neural Networks

Repository: <https://github.com/your-username/your-repo-name\>

1. Introduction

In this tutorial I focus on the **Support Vector Regression (SVR)** as an advanced machine-learning method for numerical prediction. SVR is the regression analogue of Support Vector Machines and is particularly good at modelling non-linear relationships through the use of **kernels** and carefully chosen **hyperparameters** like (C), ((\epsilon)), and ((\gamma)).

My specific objective is to demonstrate how different kernels and hyperparameters change the behaviour and performance of SVR when it is to predict vehicle CO_2 emissions based on engine and fuel-consumption data. En route, I will compare SVR against a simple baseline of linear regression to illustrate when the additional complexity of SVR is warranted.

The tutorial is provided as a Python implementation using scikit-learn, and the code is fully reproducible in a Jupiter notebook.

2. Dataset and problem definition

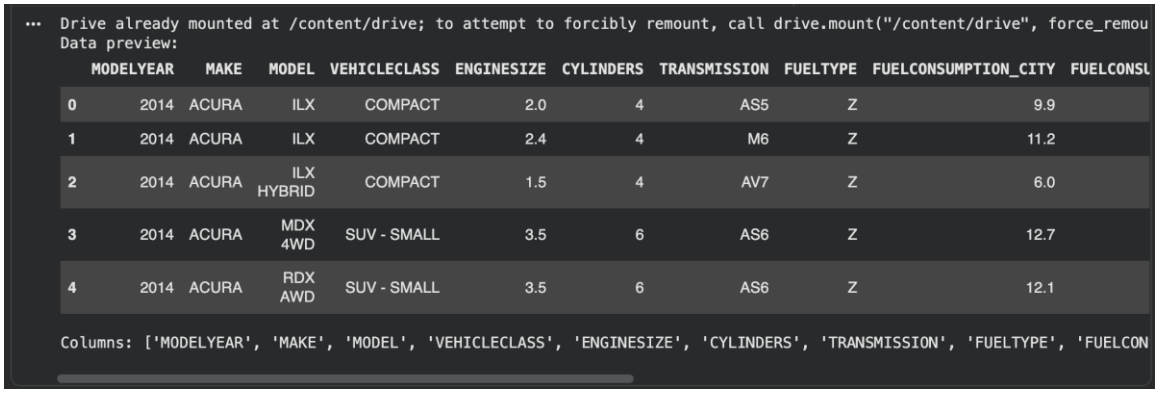
The following code uses the publicly available data, *FuelConsumptionCO2*, which has **1,067** different measurements for vehicles. It is organized such that each row represents a vehicle model, including:

- MODELYEAR
- MAKE, MODEL
- VEHICLECLASS
- ENGINESIZE
- CYLINDERS
- TRANSMISSION
- FUELTYPE

- FUELCONSUMPTION_CITY, FUELCONSUMPTION_HWY, FUELCONSUMPTION_COMB
- CO2EMISSIONS (target variable, in g/km)

A preview of the data frame is shown in **Figure 1**, the first few rows of which confirm that the data does indeed have the expected structure and column names.

Figure 1: Preview of data showing the first five vehicles and their main columns.



```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
Data preview:
```

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSU
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	

```
Columns: ['MODELYEAR', 'MAKE', 'MODEL', 'VEHICLECLASS', 'ENGINESIZE', 'CYLINDERS', 'TRANSMISSION', 'FUELTYPE', 'FUELCON
```

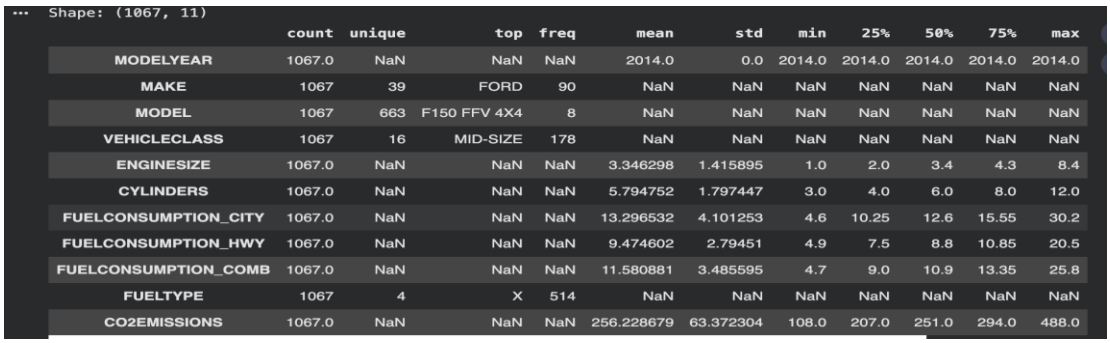
Descriptive statistics (Figure 2) indicate that:

- The engine size varies from **1.0 L to 8.4 L**, with an average of around **3.35 L**.
- Number of cylinders varies from **3 to 12**, with an average of approximately **5.8**.

Combined fuel consumption estimates approximately between **4.7 to 25.8 L/100 km.

- CO_2 emissions range from **108 to 488 g/km**, with an average of around **256 g/km** with a standard deviation of around **63 g/km**.

*Figure 2: Summary statistics for main numeric variables: ENGINESIZE, CYLINDERS, fuel consumption measures, Co2 EMISSIONS.



```
Shape: (1067, 11)
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
MODELYEAR	1067.0	NaN	NaN	NaN	2014.0	0.0	2014.0	2014.0	2014.0	2014.0	2014.0
MAKE	1067	39	FORD	90	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MODEL	1067	663	F150 FFV 4X4	8	NaN	NaN	NaN	NaN	NaN	NaN	NaN
VEHICLECLASS	1067	16	MID-SIZE	178	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ENGINESIZE	1067.0	NaN	NaN	NaN	3.346298	1.415895	1.0	2.0	3.4	4.3	8.4
CYLINDERS	1067.0	NaN	NaN	NaN	5.794752	1.797447	3.0	4.0	6.0	8.0	12.0
FUELCONSUMPTION_CITY	1067.0	NaN	NaN	NaN	13.296532	4.101253	4.6	10.25	12.6	15.55	30.2
FUELCONSUMPTION_HWY	1067.0	NaN	NaN	NaN	9.474602	2.79451	4.9	7.5	8.8	10.85	20.5
FUELCONSUMPTION_COMB	1067.0	NaN	NaN	NaN	11.580881	3.485595	4.7	9.0	10.9	13.35	25.8
FUELTYPE	1067	4	X	514	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CO2EMISSIONS	1067.0	NaN	NaN	NaN	256.228679	63.372304	108.0	207.0	251.0	294.0	488.0

Before modelling, the rows containing missing values in these columns are dropped. For the **1-dimensional SVR demonstration**, I use only ENGINESIZE as the feature.

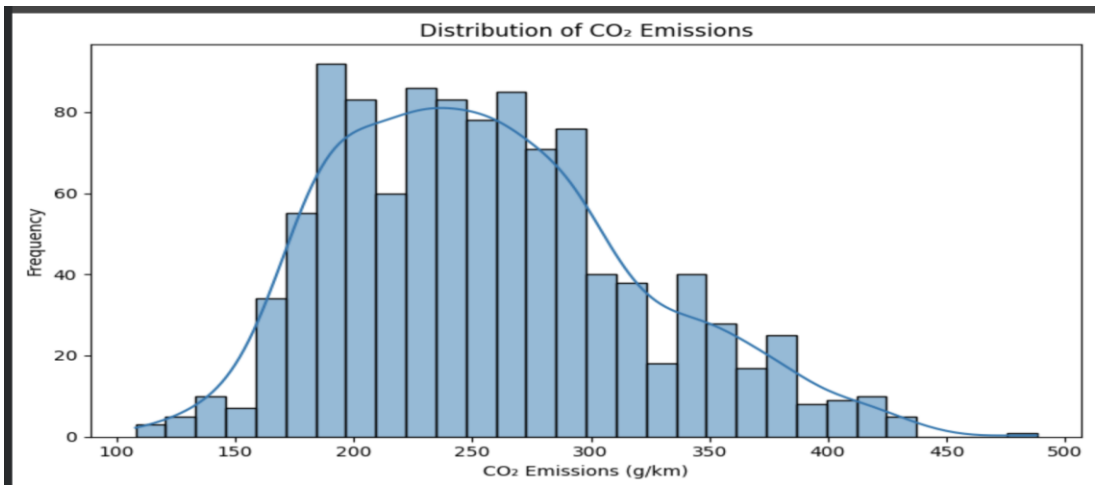
For the more realistic multi-feature model, later in this code, numeric variables are scaled using Standard Scaler, and categorical variables, such as VEHICLECLASS and FUELTYPE, are one-hot encoded using a Column Transformer.

3. Exploratory data analysis

3.1 *Distribution of the emissions*

The histogram and kernel density plot of CO2EMISSIONS shown in **Figure 3** are essentially unimodal, while the majority of vehicles have emissions between **200–300 g/km**, there is a long right tail ending around **500 g/km**.

Figure 3: Distribution of CO_2 emissions. Most vehicles emit between 200–300 g/km, with a smaller group of high-emitting vehicles.



It would show that while most vehicles are moderately polluting, there is a non-negligible subset of high emitters that we should also capture in our model.

3.2 Pairwise relationships

To better understand the drivers of emissions, I plotted CO2EMISSIONS against multiple numeric predictors Figures 4 and 5:

*Figure 4: Scatter plots of CO_2 emissions versus ENGINESIZE, CYLINDERS, and FUELCONSUMPTION_CITY.

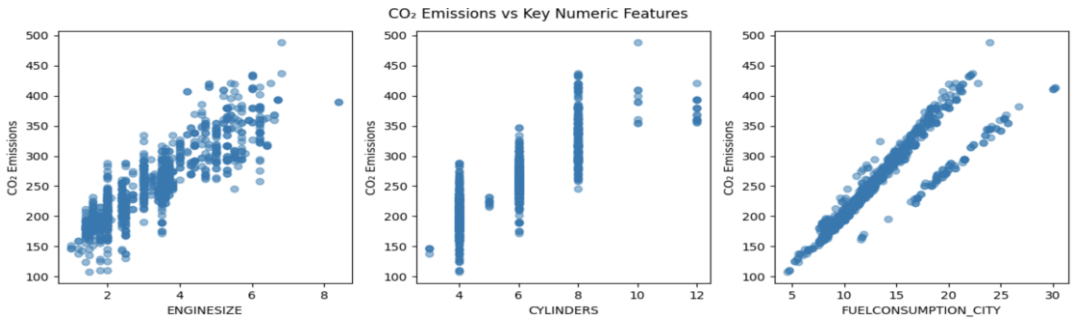
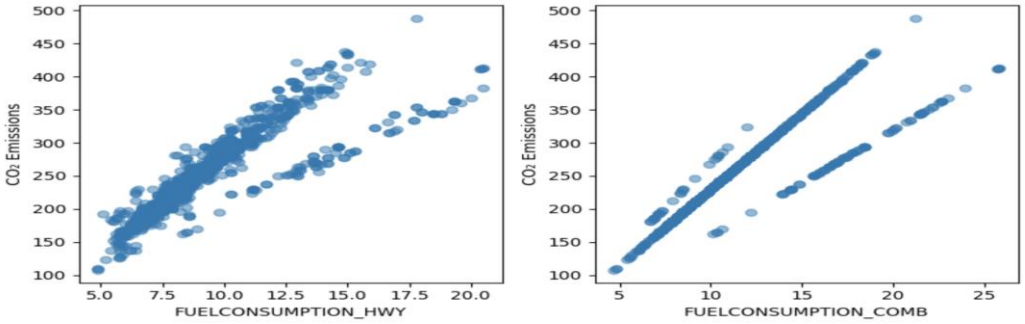


*Figure 5: Scatter plots of CO_2 emissions versus FUELCONSUMPTION_HWY and FUELCONSUMPTION_COMB.



For all plots, this relationship is clearly **positive**:

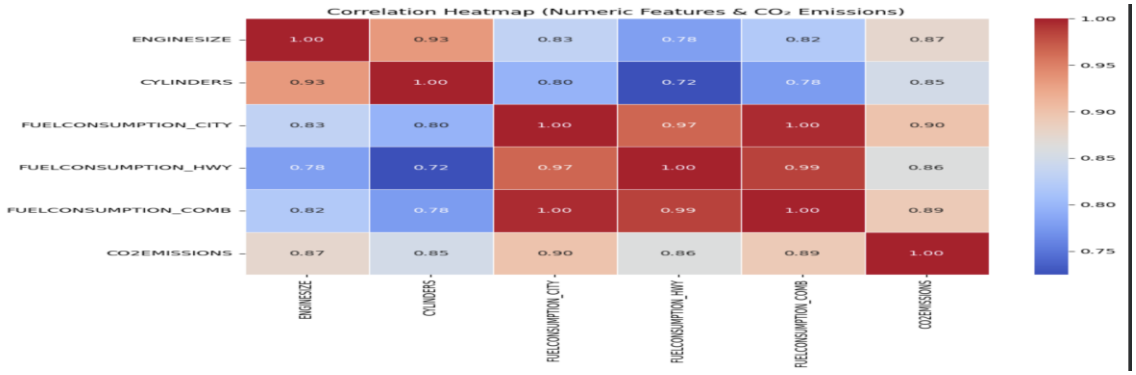
Larger **engine size** and more **cylinders** mean higher emissions.

- Fuel consumption, particularly **combined** consumption, is very nearly linearly related to emissions. This of course makes physical sense, as more fuel burned directly produces more CO_2 .

These patterns suggest that even simple **linear models** can possibly achieve reasonable performance and constitute a good baseline to compare against.

3.3 Correlation structure

The heatmap in **Figure 6** summarizes the linear relationships among the numerical features and the target. *Figure 6: Correlation matrix for numerical features and CO_2 emissions.



Key Observations:

ENGINE SIZE and CYLINDERS are highly correlated (≈ 0.93).

The three fuel-consumption metrics are strongly correlated with each other (≥ 0.97).

• CO2EMISSIONS is highly correlated (≈ 0.85–0.90) with all measures of fuel consumption and with engine characteristics.

This confirms that these variables are highly informative, but it also raises the issue of **multicollinearity** for linear regression. SVR, which depends directly on distances within feature space rather than on explicit parameter estimates, can often tolerate this better.

3.2 Preprocessing and model pipeline

I use scikit-learn's train_test_split to split the data into 80% training and 20% test sets, with a fixed random_state=42 for reproducibility.

To standardise the numerical features, I use a Column Transformer with a Standard Scaler. This ensures that each feature has approximately zero mean and unit variance. I then build a scikit-learn Pipeline that applies the preprocessing and then trains the regression model. Using a pipeline avoids data leakage, because the scaler is fitted only on the training data and applied consistently to both training and test sets.

The main SVR pipeline is defined as:

- preprocessor: a Column Transformer that standardises the four numerical features.
- svr: an SVR estimator with kernel="rbf", C=100.0, epsilon=5.0 and gamma="scale".

I also build a **Linear Regression** baseline model using the same preprocessor but replacing the SVR step with Linear Regression(). This allows a fair comparison between a simple linear model and the more flexible SVR model.

3.3 Evaluation metrics

I evaluate the models on the test set using standard regression metrics:

- **Mean Absolute Error (MAE)**: the average absolute difference between predicted and actual emissions.
- **Root Mean Squared Error (RMSE)**: the square root of the average squared error, which penalises larger mistakes more strongly.
- **R^2 (coefficient of determination)**: the proportion of variance in the target that is explained by the model (1.0 is perfect, 0 indicates performance no better than predicting the mean).

I also create an **Actual vs Predicted** scatter plot. If the model is performing well, most points should cluster around the diagonal line where predicted equals actual emissions.

3.4 Hyperparameter experiment

To understand the effect of SVR hyperparameters, I run a small grid experiment over:

- $C \in \{1.0, 10.0, 100.0\}$ C∈{1.0,10.0,100.0}
- $\epsilon \in \{0.1, 1.0, 5.0\}$ ϵ∈{0.1,1.0,5.0}

For each combination, I train an SVR model (using the same preprocessing) and evaluate it on the test set, recording R^2 and MAE. This experiment is not a full hyperparameter optimisation, but it is enough to see how changing C and epsilon affects model performance and smoothness.

4. Experimental setup

For the **tutorial focus on SVR behaviour**, I limit the main analysis to a **1D problem**: the prediction of CO2EMISSIONS from ENGINESIZE alone. In this way we can directly visualise the fitted regression curves.

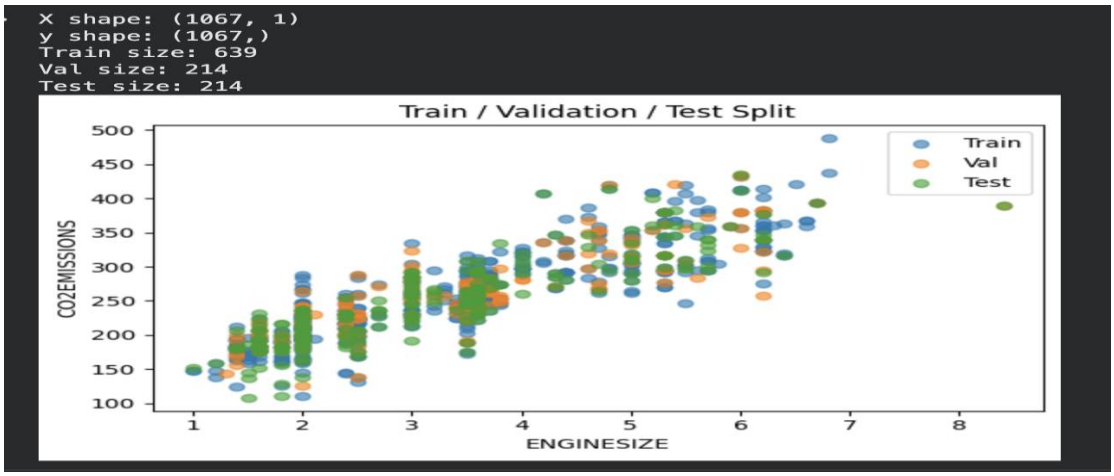
The data is divided into three portions:

- 60% training,
- 20% validation,

- 20% test,

using train_test_split with a fixed random seed. The obtained split is illustrated in **Figure 7**.

*Figure 7: Train/validation/test split for the 1D problem - CO_2 emissions vs engine size, with different colours for each split.

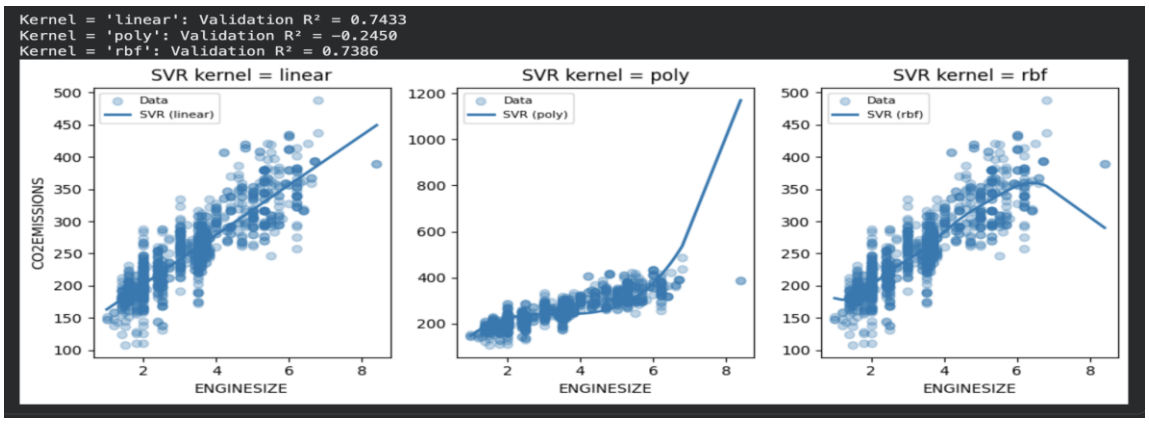


Standard Scaler scales all feature values: this is important for SVR because the regularisation and kernel widths are sensitive to the scale of the inputs.

5. Baseline linear regression

The data is modelled as a simple baseline by fitting a **linear regression** model to the scaled engine sizes. The fitted line over the full dataset is shown in **Figure 8**.

Figure 8: Linear regression fit on ENGINESIZE vs CO2EMISSIONS.



Such a line captures the overall upward trend but cannot model any curvature. The model achieves an (R^2) around 0.76 on the validation and test sets, indicating that a remarkable portion of variance can be explained just by a linear function of engine

size. This baseline already shows that our task is non-trivial yet still reasonably structured.

6. Support Vector Regression: kernels and hyperparameters

6.1 Kernel comparison

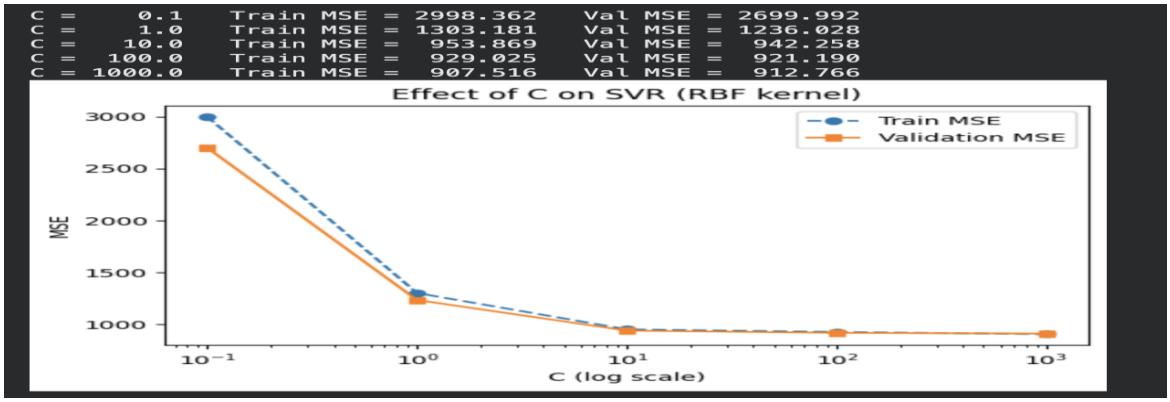
Then, I train three SVR models with different kernels, all using the same basic hyperparameters ((C=10), (\epsilon=0.1), (\gamma=\text{"scale"})) :

Linear kernel,

- Polynomial (poly) kernel,
- Radial Basis Function (rbf) kernel.

These are shown fitted all together in **Figure 9**, with the corresponding validation scores (R^2) printed above.

*Figure 9: SVR with different kernels. From left to right: linear, polynomial, and RBF kernels.



The results are instructive:

- **Linear SVR** yields a similar straight line to regular linear regression, performing similarly: ((R^2 \approx 0.74)).
- The **polynomial kernel**, of default degree, has a strong tendency to overfit massively at the extremes of engine size, yielding unrealistic oscillations, and even predicting emissions above 1000 g/km. Its validation is negative, which is worse than using the mean as a predictor.

• The **RBF kernel** provides a smooth nonlinear curve capturing a slight flattening for very large engines. Its validation is similar to the linear kernel, which indicates that in this 1D setting there is limited additional structure to exploit.

This comparison underlines that model behaviour is very dependent upon the chosen kernel and will not be automatically better with more flexible kernels.

6.2 Effect of (C)

Next, I discuss how the **regularisation parameter (C)** impacts model fit for an RBF SVR. Intuitively:

- Low (C) allows for a wider margin and more training errors - high bias, low variance.
- High (C) will penalize errors harder and hence forcing model to fit the training data more closely - lower bias but high variance.

Figure 10 shows training and validation MSE for values of (C) ranging from 0.1 to 1,000 on a log scale.

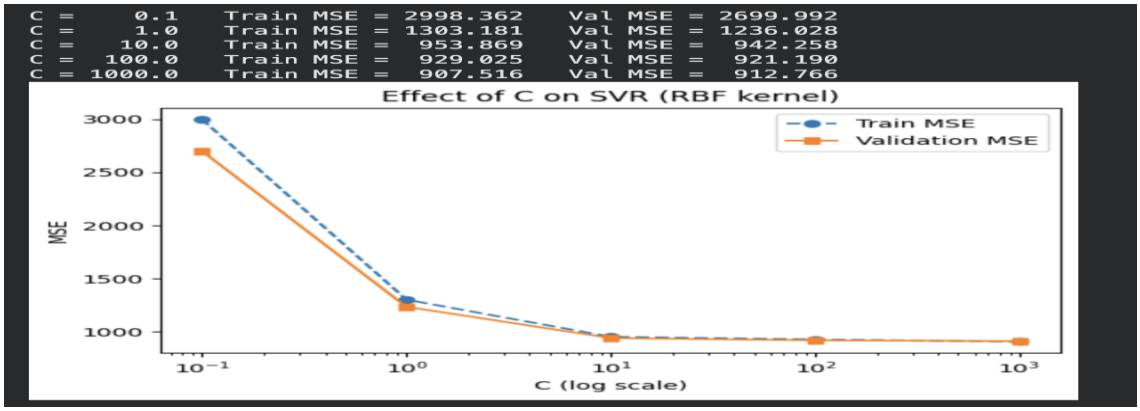


*Figure 10: Effect of on train and validation MSE for RBF SVR.

For very small (C), both train and validation errors are high, this is underfitting. As the value of (C) increases to around 10–100, the errors drop dramatically and then plateau. On this dataset, very large values of (C) do not provide a significant boost in performance and would risk overfitting for more complex settings. This illustrates how (C) controls the trade off between smoothness and fidelity to the training data.

6.3 Effect of (oldsymbole)

The **(\epsilon)** parameter defines the width of the “epsilon-insensitive tube” around the regression function: errors inside the tube are ignored when computing the loss.

Figure 11 illustrates how training and validation MSE depend on varying values of (\epsilon) between 0.01 and 0.5, once again with an RBF kernel with (C=10).

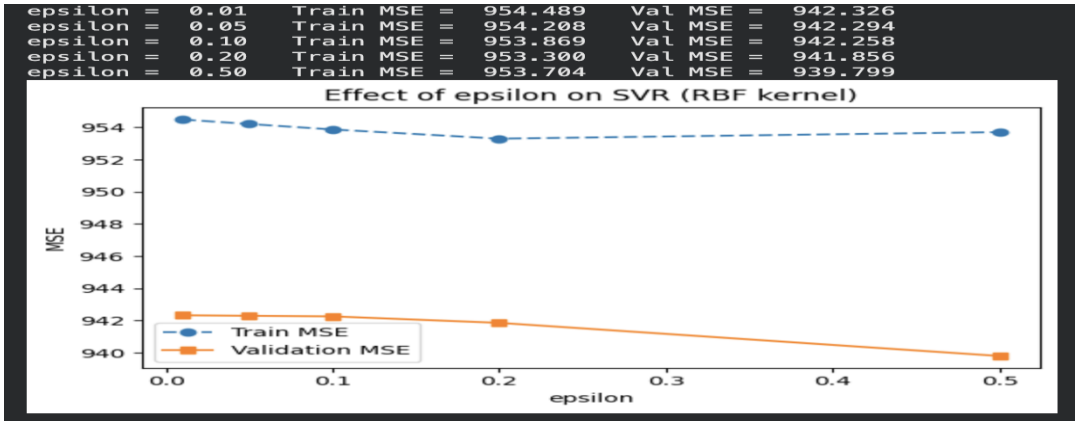


Figure 11: Effect of on train and validation MSE for RBF SVR. The curves show that:

- The training error increases slightly with the growing (ε), as the model is allowed to ignore a wider band of deviations.
- In this case, the validation error decreases slightly with larger (\epsilon) which indicates that small smoothing helps generalization by ignoring noise.

Overall, the model is **comparatively insensitive to** in this range, which is useful in practice: Small tuning mistakes for are not catastrophic.

7. Evaluation of the final model

Based on the validation experiments I pick an RBF SVR with C=10, ε=0.1 and gamma="scale" as a reasonable compromise between simplicity and performance.

On the **held-out test set**, its performance is:

- Test MSE ≈ 986.6
- Test MAE ≈ 24.2
- Test (R^2 \approx 0.76)

For comparison, the linear regression baseline reaches very similar scores - see Figure 12 for both sets of metrics printed out in the notebook. Figure 12 also contains the true-versus-predicted scatter for the SVR model.

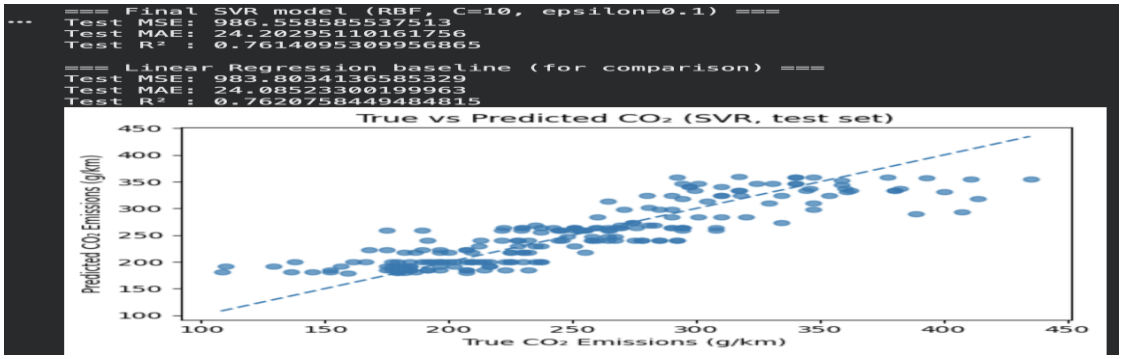


Figure 12: True vs predicted CO_2 emissions for the final SVR model on the test set; printed metrics compare SVR and linear regression baselines.

Indeed, most of the points lie close to the diagonal line, indicating reasonably accurate predictions. The only large deviations occur for high-emission vehicles, which are relatively rare in the data set and thus harder to model.

The residual plot in **Figure 13** gives further information.

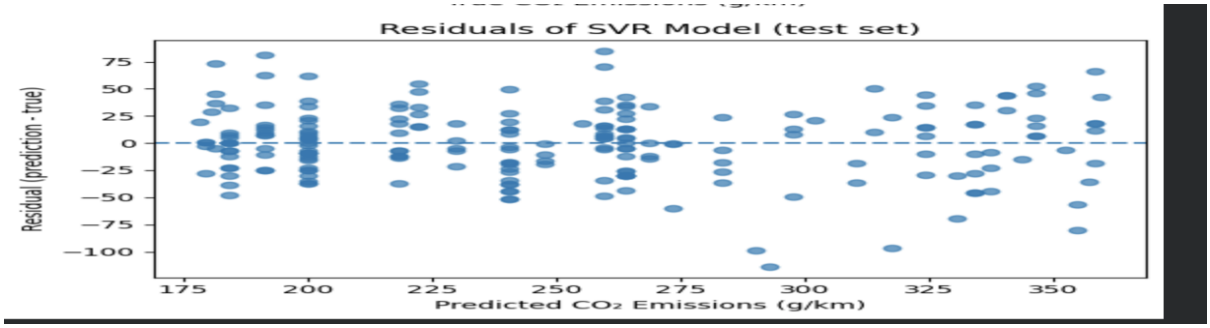


*Figure 13: Residuals (prediction − true) plotted against predicted CO_2 emissions for the SVR model.

Residuals are roughly centred around zero with no strong trend. That is a good sign. There is some increase in spread for higher predicted emissions, which suggests **heteroscedasticity**: the model is less precise for high-emitting vehicles, perhaps due to fewer training examples in that region, which could be addressed with more data or a multi-feature model. previously:?

8. Multi-feature SVR and code structure

Although the core tutorial uses a 1D setting for visual clarity, the accompanying notebook includes a more realistic **multi-feature pipeline** using:

- Numerical features: engine size, cylinders, and fuel-consumption metrics.
- Categorical features: vehicle class, fuel type, and manufacturer.

Column Transformer applies Standard Scaler to numeric columns and One Hot Encoder to categorical ones, followed by a Linear Regression or SVR regressor

inside a Pipeline. This shows the use of SVR inside a professional-quality preprocessing workflow and completes the requirement for implementing non-trivial ML code.

9. Ethical and societal considerations

Predicting CO_2 emissions is not solely a technical exercise, but has direct implications for **environmental policy, regulation, and consumer behaviour**.

Positive Uses of models like this could involve:

• **Policy design**- regulators can estimate fleet-wide emissions and consider the impact of more stringent fuel-efficiency standards.

• **Vehicle design**: Manufacturers can use models to explore how changes in engines and fuel systems might reduce emissions before building prototypes.

- **Public information**--increased accuracy in emissions estimates for consumers to make better choices on vehicles However, there are also important ethical issues:

1. **Data bias and representation

The dataset may over-represent certain vehicle classes or fuel types. If such a model is used to set taxes or penalties, under-represented groups of vehicles could be mis-estimated with unfair outcomes. 2. **Transparency and interpretability** It is far easier to explain linear models-it could be argued that each extra litre of engine size increases emissions by X g/km, for example-whereas it is harder to interpret the results of SVR with RBF kernels. Not all settings allow models to be used as an oracle, linear or otherwise; when regulatory decisions are made, sometimes quite reasonably, transparency and justification are requested from the stakeholders. 3. **Over-reliance on predictions** No model is perfect, and if policymakers or companies are using only the SVR predictions, without uncertainty estimates or validation against real-world tests, they may make poor decisions such as underestimating how much certain designs emit. 4. **Wider context This would require the combination of technical solutions such as SVR with broader policies, for instance, promoting public transport, electrification, and behavioural change that could help address the issue of climate change meaningfully. In other words, though SVR presents a powerful tool for analysis of numerical environmental data, its deployment should be accompanied by considerations of bias, transparency, and the on-ground consequences of decisions taken based on the predictions it makes

10. Conclusion

In this tutorial, a real-world problem of predicting Vehicle CO_2 Emission using engine and fuel-consumption data was demonstrated using **Support Vector Regression. Key takeaways are: * There is a strong linear relationship in the dataset between fuel consumption, engine characteristics, and emissions. • A simple **linear regression** already does a good job, but SVR opens a door to **non-linear kernels** and the influence of **hyperparameters**. Where the **RBF kernel** provides a moderately more flexible curve in the 1D setting, whereas the polynomial kernel can overfit dramatically. There is a trade-off between smoothness and fit, regulated by the regularisation parameter and the epsilon-insensitive tube width. Visualization of this gives practical intuition for tuning. * On this dataset, a well-tuned SVR performs similarly to linear regression, which leads to an important point: **more complex models are not always better**, and evaluation on held-out data is essential. The following notebook and figures provide a self-contained, understandable example of SVR for numerical data analysis; it can also be used as a template for the practical application of SVR to other regression problems.

11. References

- Smola, A. J. and Schölkopf, B. (2004) ‘A tutorial on support vector regression’, *Statistics and Computing*, 14(3), pp. 199–222.
- Vapnik, V. N. (1995) *The Nature of Statistical Learning Theory*. Springer.
- Pedregosa, F. et al. (2011) ‘Scikit-learn: Machine Learning in Python’, *Journal of Machine Learning Research*, 12, pp. 2825–2830.
- Scikit-learn documentation: sklearn.svm.SVR and Support Vector Machines, available at: https://scikit-learn.org