Cars 24 Problem Overview

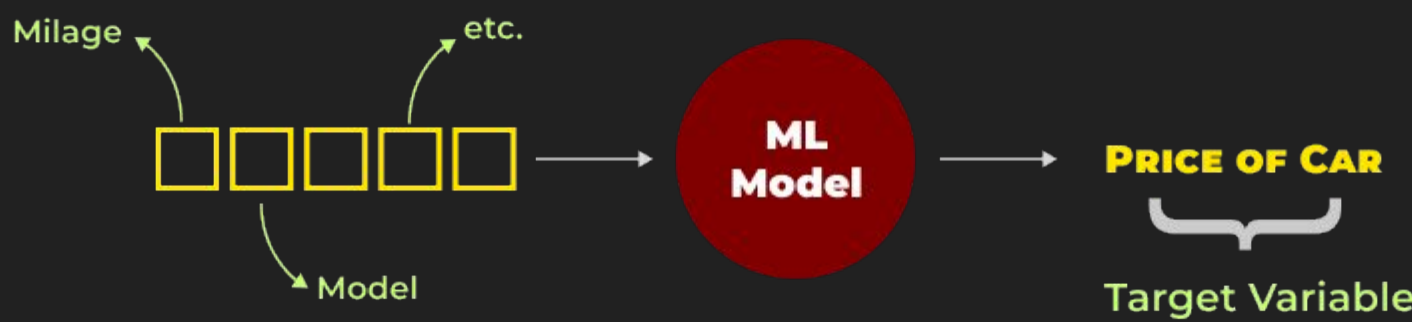
Data scientist at **Cars 24** → Sells pre-owned cars

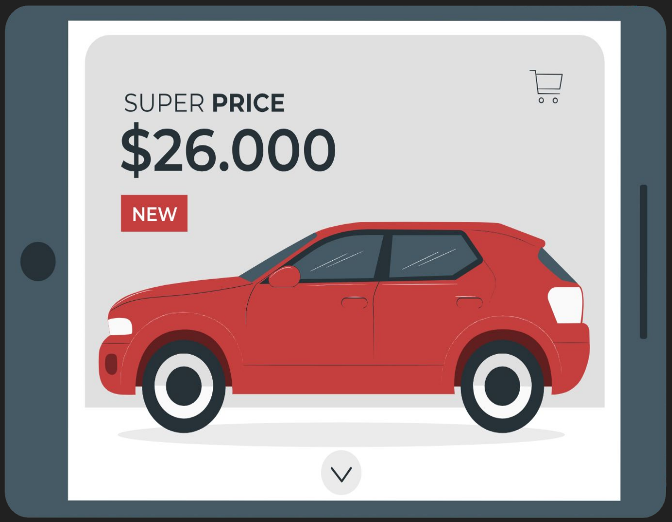
→ To automate pricing the old car

Given car features

(Make, Model Mileage , Odometer, Service History etc.) → **Predictors / Features**

Train an ML Mode such that





- We have target in training data
  - SUPERVISED
- Target variable is continuous
  - REGRESSION

Look at the Data . .

	selling_price	km_driven	mileage	engine	max_power	age	make	model	In
0	1.20	120000	19.70	796.0	46.30	11.0	MARUTI	ALTO STD	
1	5.50	20000	18.90	1197.0	82.00	7.0	HYUNDAI	GRAND I10 ASTA	
2	2.15	60000	17.00	1197.0	80.00	13.0	HYUNDAI	I20 ASTA	
3	2.26	37000	20.92	998.0	67.10	11.0	MARUTI	ALTO K10 2010-2014 VXI	
4	5.70	30000	22.77	1498.0	98.59	8.0	FORD	ECOSPORT 2015-2021 1.5 TDCI TITANIUM BSIV	

Noticed any issue with the data ?

1. Make and model column have lots of categories
2. All columns are in different ranges e.g. age and km_driven

Revision - Target Variable Encoding

Make: **41 categories**

Using One Hot Encoding will result in a lot of dimensions -> **curse of dimensionality**

Hence, Target Encoding

Make	Selling Price
Maruti	1.2
Hyundai	5.5
Hyundai	2.15
Hyundai	2.26
Ford	5.70

Target Encoding

Make	Selling Price
1.2	1.2
3.3	5.5
3.3	2.15
3.3	2.26
5.7	5.70

MEAN

Maruti - **1.2**

Hyundai - **3.3**

Ford - **5.70**

Replace

Target encoding replaces the categories with a number representing the average target value associated with each category.

Line of Code

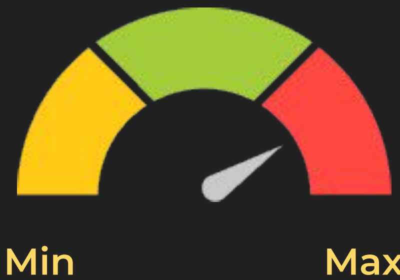
```
df['make'] = df.groupby('make')['selling_price'].transform('mean')

df['model'] = df.groupby('model')['selling_price'].transform('mean')
```

Revision - Scaling the Data

$$X_{Scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Min - Max Scaler



Km_Driven
10,000
5000
2000

Scaling

Km_Driven
1
0.375
0

$$X_{min} = 2000$$

$$X_{max} = 1000$$

Note:

Since all the features are in different ranges, we need to scale the data.

Later we will see why is scaling important

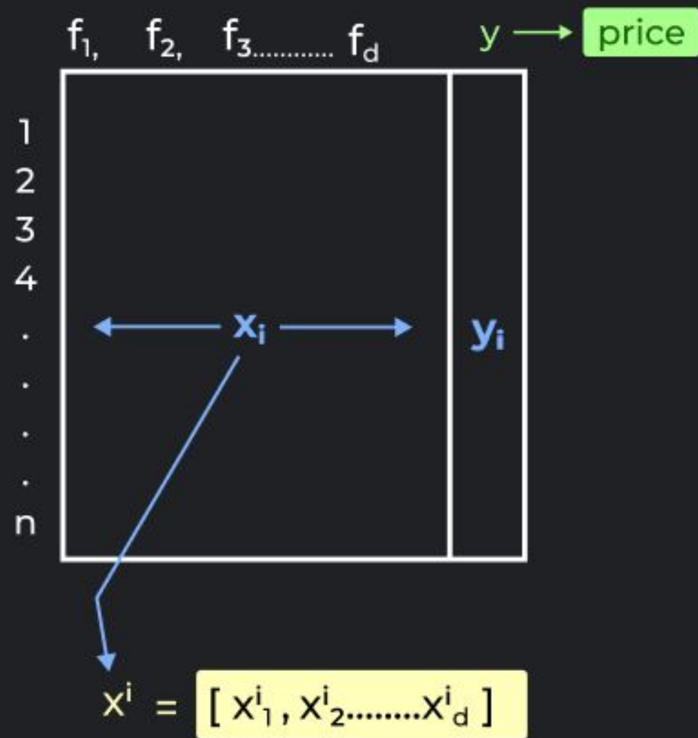
Line of Code

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

Data Notation



(m,d)
↓
X

m → no. of samples

d → no. of features

n → no. of classes

$i^{[th]}$ $sample \rightarrow x^{[i]}$

$j^{[th]}$ $feature \rightarrow x_j$

True o/p ⟶ $y^{[i]}$

Predicted o/p ⟶ $\hat{y}^{[i]}$

Don't get intimidated by the terms,
we will go through all of them as we proceed with this module.

Goal of generalization in ML

Assume you had m samples $\{X^i, X^i\}^m$

Using Historical data we train ML Model

$x^i \xrightarrow{\text{ML Model}} \hat{y}^i$ — Predicted Value

Ideally, $\hat{y} \approx y$ — Original Price / Ground Truth

If for m samples at an average $\boxed{\hat{y} \approx y}$ → GOOD TRAINING

But,

Goal: Predict well on New sample.

$$x^i_{\text{New}} \xrightarrow{\text{ML Model}} \hat{y}^i$$

Problem??

How do we know if our model is good in predicting for NEW sample ?

$\hat{y} \approx y \longrightarrow$ UNKNOWN

Solution:

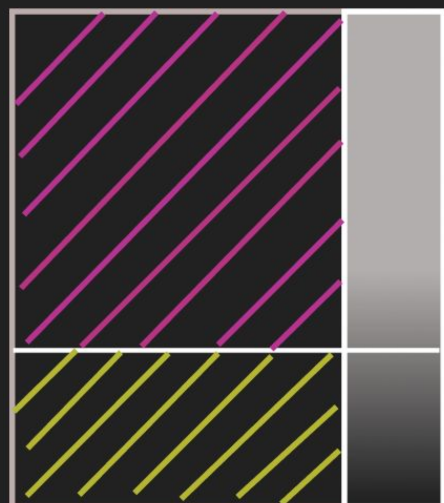
Because we don’t know the ground truth (y) for New Sample

DO NOT use whole data for TRAINING

Typically,

80%

20%



Use this for training the data

Use this as a proxy for unseen data

‘y’ known

Check if: $\hat{y} \approx y$

For evaluating model performance

Phases for Model Development

Training - use train split

$\hat{y} \approx y$ To check if model is learning

Testing- use test split

$\hat{y} \approx y$ To check if model is generalising

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=1)
```

Linear Regression Intuition - Single variable

Let's take one feature : Age

Suppose we want to predict price of car from it's age

How do you think price of car change with it's increasing age ?

-> Decrease with age

y is a function of x

$$x^i \xrightarrow{f()} y^i$$

$$\hat{y}^i = f(x^i)$$

f () can be x+2, x^2+3, sin (x), etc

1 predictor LR = Straight line

$$y^i = mx + c$$

$$y^i = w_1 x + w_0$$

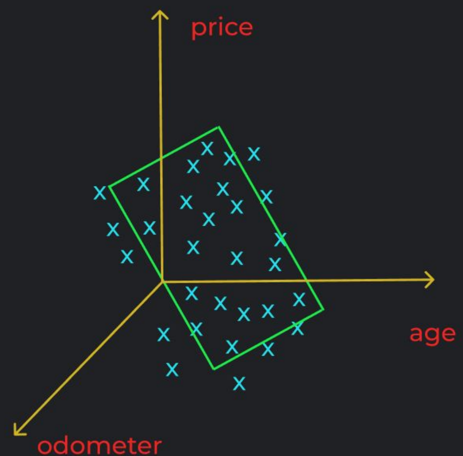
Learning?? → Estimating best values of <w> vector

Geometrically,

Linear Regression is trying to find a line such that most of the points are close to the line

Now, what if we take 2 predictors? -

age , odometer



Weightage given to x1

Bias (to be discussed later)

$$\hat{y}^i = w_1 x_1 + w_2 x_2 + w_0$$

Weightage given to x2

What would be the linear function for this case, in 3D??

A Plane

Univariate linear regression – 1 **feature** → **Straight line 2D**

Bivariate linear regression —- **2 features** → **Plane 3D**

Multivariate linear regression — **d features** → **d+1 hyperplane**

Learning in Linear Regression

Estimating best values of < w0, w1, w2, ...wd >

Which means to find the weights of best fitting line

Linear Regression using Sklearn

Univariate

Fitting the model on 1 predictor

```
X1=X[['model']]
X1_train = X_train[['model']]
X1_test = X_test[['model']]
```

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X1_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Univariate

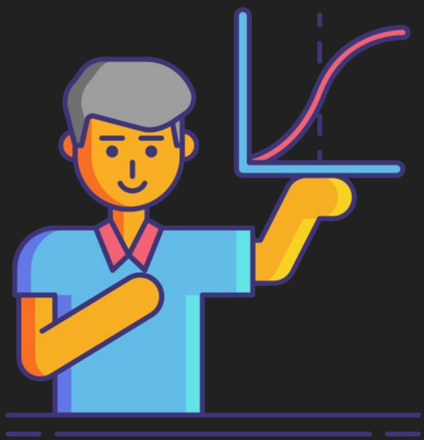
Model Weight and Bias

```
model.coef_
```

```
array([0.9967642])
```

```
model.intercept_
```

```
0.0015237505846132926
```



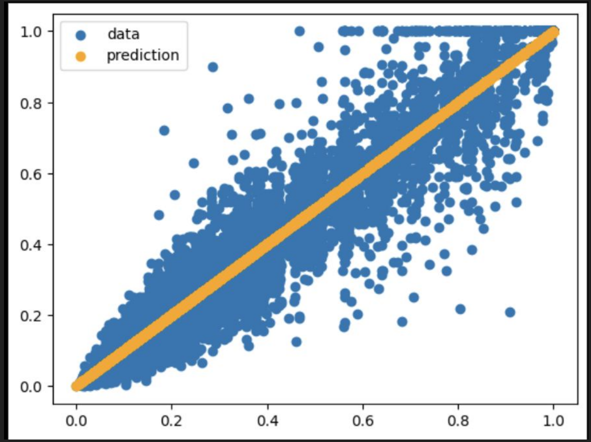
Linear Regression using Sklearn

Univariate

Plotting the predicted values

```
y_hat = model.predict(X1)

fig = plt.figure()
plt.scatter(X1,y,label='data')
plt.scatter(X1,y_hat,color='orange',label='prediction')
plt.legend()
plt.show()
```



Model : Wt. and Bias

```
model.coef_
```

```
array([0.9967642])
```

```
model.intercept_
```

```
0.0015237505846132926
```

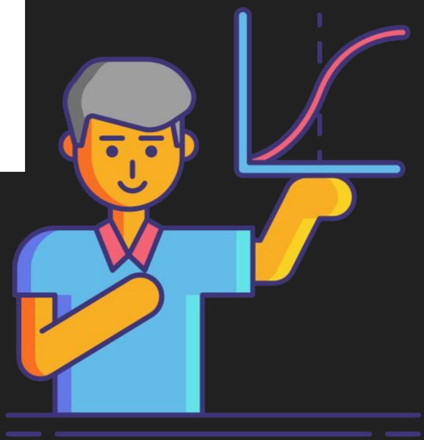
Linear Regression using Sklearn

Multivariate

Training on d features

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```



Linear Regression using Sklearn

Multivariate

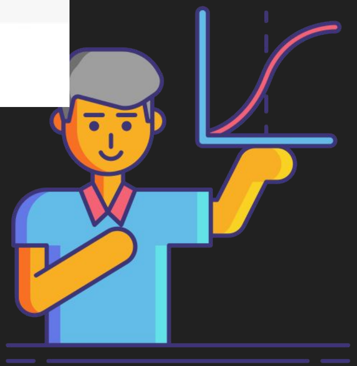
All features weight and bias

```
model.coef_
```

```
array([ 7.23382199e+11, -2.50488281e-01, -2.32558310e-01,  7.39953774e-02,
        4.68742062e-02,  7.23382199e+11,  6.61604471e-02,  8.58973267e-01,
       -7.18488746e-03, -7.03116019e-03,  6.98577387e-03,  1.32957359e-01,
        1.50077817e-02, -6.83704171e-03, -3.69616522e-03, -1.62563011e-02,
       -2.35725832e-02])
```

```
model.intercept_
```

```
-723382198910.7482
```



Is my model good?

Is there any way to measure the performance of our model?

Let's break down the problem.

Goal -> To minimize the distance between the predicted point and the actual point.

The closer the predicted value to the actual value, the better is the model.

How can we evaluate our model?

Let’s take our price vs age model:

time

y^i

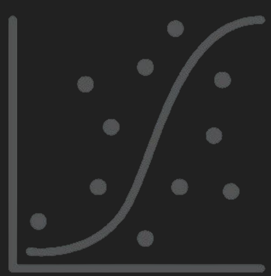
error e:

$\hat{y}^i$

predicted

$$y^i - \hat{y}^i = \text{error}$$

minimize



Aim?

Find Better Model with

Lesser Error

$$\min_{w_0, w_1} \sum_{i=1}^{m} e^i$$

Best w_0, w_1 $\longrightarrow$ $\sum_{i=1}^{m} e^i$ $\longrightarrow$ Min

Can we use $\sum_{i=1}^{m} e^i$ as a metric?

NO! WHY??

So, for multiple points..

It will be the sum of the error divided by m : $\frac{1}{m}\sum_{i=0}^{m}(\hat{y}^{(i)} - y^{(i)})$

Can we use sum of errors as a metric?? → NO

	Error	Value
1	e1	3
2	e2	4
3	e3	-7

$$\sum_{i=1}^{m} e^{i} = 0$$

Perfect model??

WRONG!!

SOLUTION ??

$$Mod -> \frac{1}{m}\sum_{i=0}^{m} |\hat{y}^{(i)} - y^{(i)}|$$

$\boxed{\text{MAE}}$ ABSOLUTE

$$Square \quad -> \frac{1}{m}\sum_{i=0}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

$\boxed{\text{MSE}}$ SQUARED

$\boxed{\text{LOW ERROR = BETTER MODEL}}$

Best among two models ?

	MSE
Model 1	21
Model 2	31

M1 IS BETTER!!

Issue with MSE or MAE,

-> it gives a value between 0 to ∞

-> value depends on the range of values predicted.

Can be in 100s, 1000s, or 10000s

How small of an MSE is good enough?

Is 1 a good mse score, or 10, or 100?

This makes a problem since there is no common scale/range to measure the performance.

What if it is a single model?

MSE/MAE → Compare n models

1 Model → How to evaluate?

Here we can use a metric called "R2 Score", or the "coefficient of determination".

It tells us how well a regression model fits the data.

R^2 score

Let's plot the pt.s between x and y for model M1

What will be worst model that we can built?

Mean model

- Mean model returns mean of yi's as predicted value every time.

SS_residual : **sum of squared errors for the model that we built**

SS_total : **sum of squared errors if the model was mean model.**

R^2 score

$$\frac{SS_{res}}{SS_{total}} = \frac{\frac{1}{m}\sum_{i=0}^{m}(y^{(i)} - \hat{y})^2}{\frac{1}{m}\sum_{i=0}^{m}(y^{(i)} - \bar{y})^2}$$

Predicted

Mean

Generally

0
Bad → **1**
Good

$\frac{SS_{RES}}{SS_{TOTAL}}$

SMALL → GOOD MODEL

LARGE → BAD MODEL

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOTAL}}$$

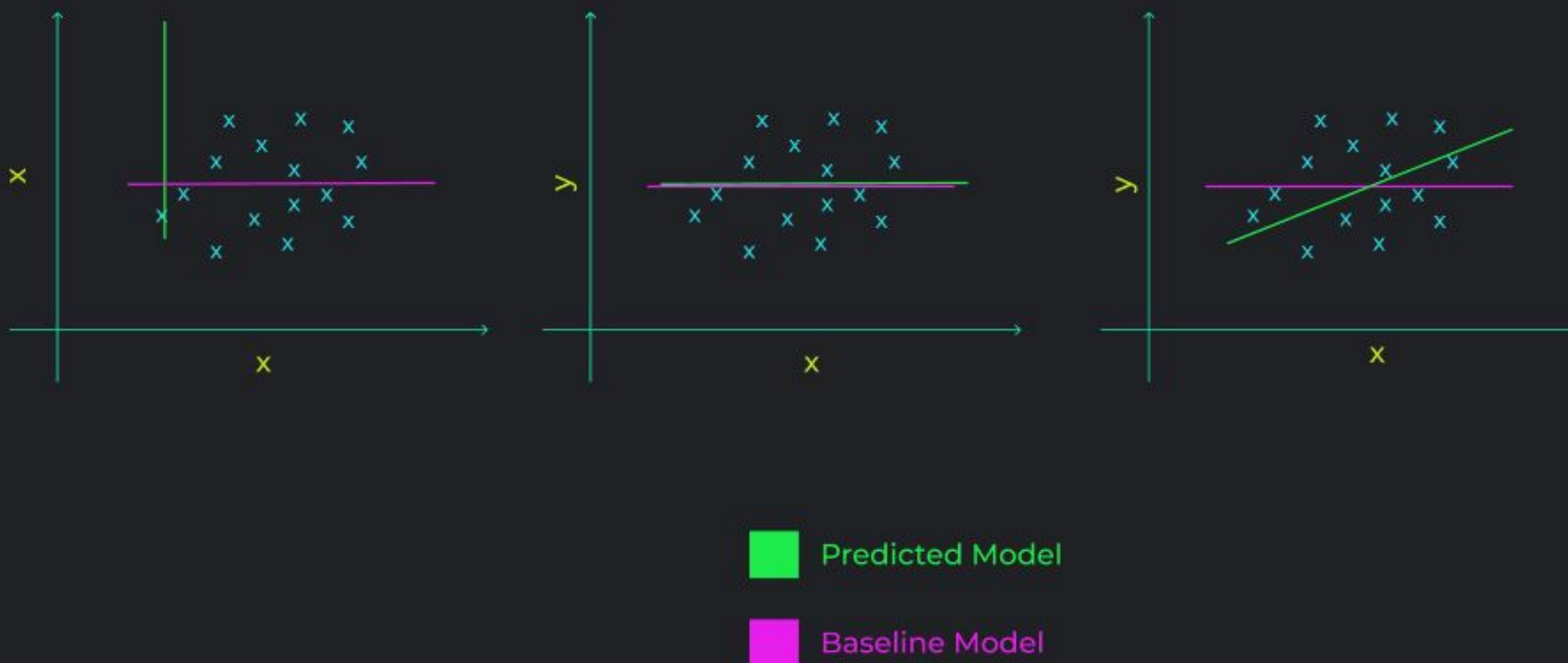
R² score

- ∞	1	0
$\frac{SS_{RES}\uparrow}{SS_{TOTAL}\downarrow} \longrightarrow \infty$	$\frac{SS_{RES}}{SS_{TOTAL}} = 1$	$\frac{SS_{RES}\downarrow}{SS_{TOTAL}} \longrightarrow 0$
Worse than baseline	Equally bad as baseline	Much better than baseline

Extremely Bad Model

Mean Model

Great Model



Range for r2 score = $[-\infty, 1]$

Model Interpretability

We know,

$$\text{Model } \hat{y}^i = W_1 X_1 + W_2 X_2 + W_3 X_3 + \dots\dots\dots + W_d X_d + W_0$$

Suppose

$$\text{Model } \hat{y}^i = w_0 + w_1 x_1 + (-10000)\ \text{age} + (10)\ \text{engine} \quad + w_n x_n$$

How to understand these weight values?

Cases as per sign

1. -ve weight of feature

wt. ↑ → $\hat{y}$ ↓

wt. ↓ → $\hat{y}$ ↑

2. +ve weight of feature

wt. ↑ → $\hat{y}$ ↑

wt. ↓ → $\hat{y}$ ↓

3. Wt = 0

wt. ↑ or wt. ↓ → No effect

Cases as per magnitude

Let's take e.g of age & engine

Age wt. = -10000

engine wt. = 10

AGE → AGE + 1 $\hat{y} \longrightarrow \hat{y} - 10000$

ENGINE → EG + 1 $\hat{y} \longrightarrow \hat{y} + 10$

LARGER THE ABSOLUTE VALUE, MORE IMPORTANT THE FEATURE

So, is age >>>>> engine imp??

Can’t be true!!

WHY??

SCALE IS DIFFERENT, UNIT IS DIFFERENT

AGE - [1, 15] Years

ENGINE - [500, 5000] cc

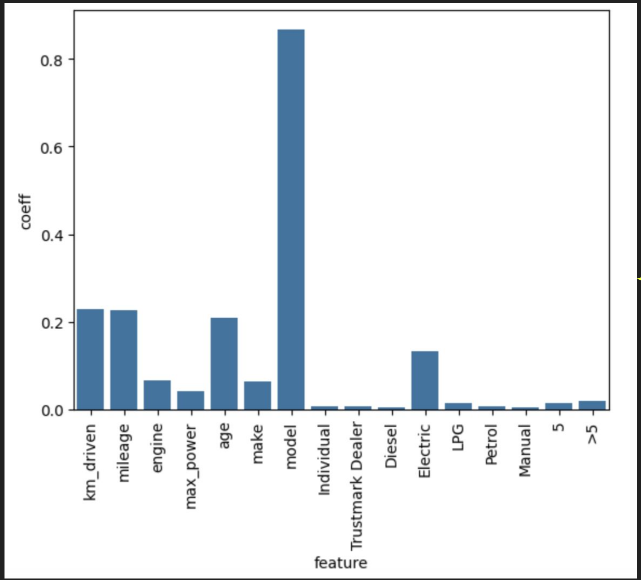
Solution

Feature Scaling

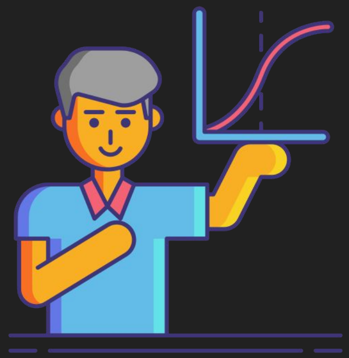
Bar Plots of Feature Importances

```
import seaborn as sns

imp = pd.DataFrame(list(zip(X_test.columns,np.abs(model.coef_))),
                   columns=['feature', 'coeff'])
sns.barplot(x='feature', y='coeff', data=imp)
plt.xticks(rotation=90)
```



“Model” is the most important



Which feature is most / least important??

```
X_test.columns[np.argmax(np.abs(model.coef_))]
```

'model'

```
X_test.columns[np.argmin(np.abs(model.coef_))]
```

'Manual'

“Year” —- Most important

“Manual” —-- Least important