20/2/2024

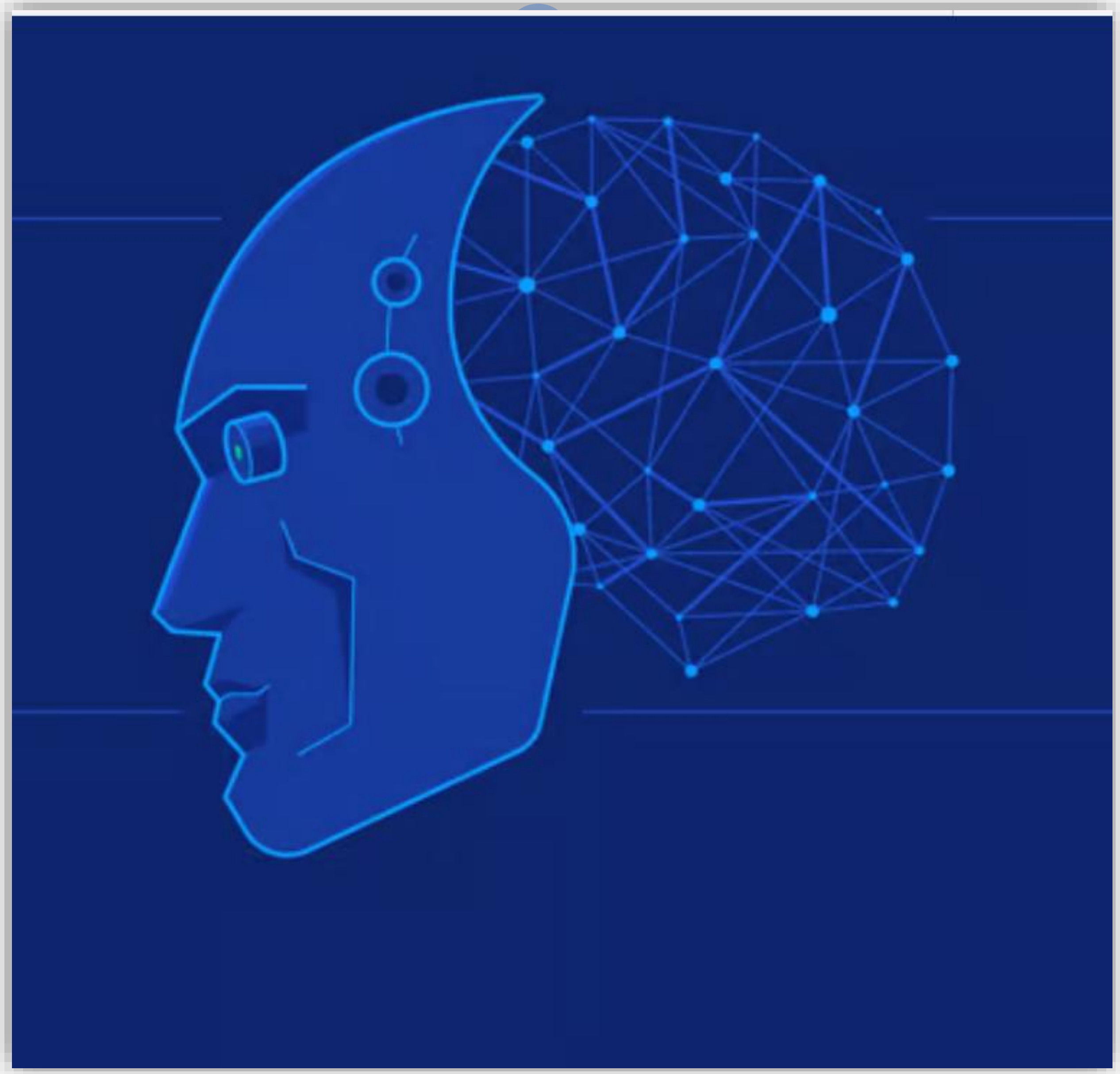
Predicting California Housing

February 20, 2024

Khaledgama4@gmail.com

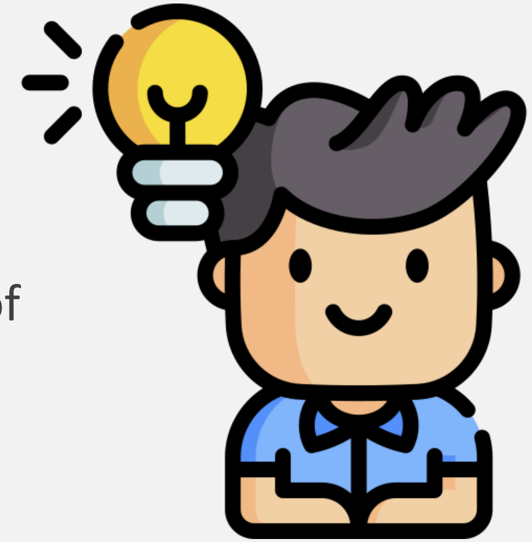
Agenda

- Problem definition
- Data
- Modelling
- Evaluation
- Tuning a model (improving it)



Problem Definition

Given parameters about Housing in California, can we predict the price of average house value for California districts?



Data interpretation

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Check Nulls & Datatypes

```
housing_df.dtypes
```

```
MedInc        float64
HouseAge      float64
AveRooms      float64
AveBedrms     float64
Population    float64
AveOccup      float64
Latitude      float64
Longitude     float64
target        float64
dtype: object
```

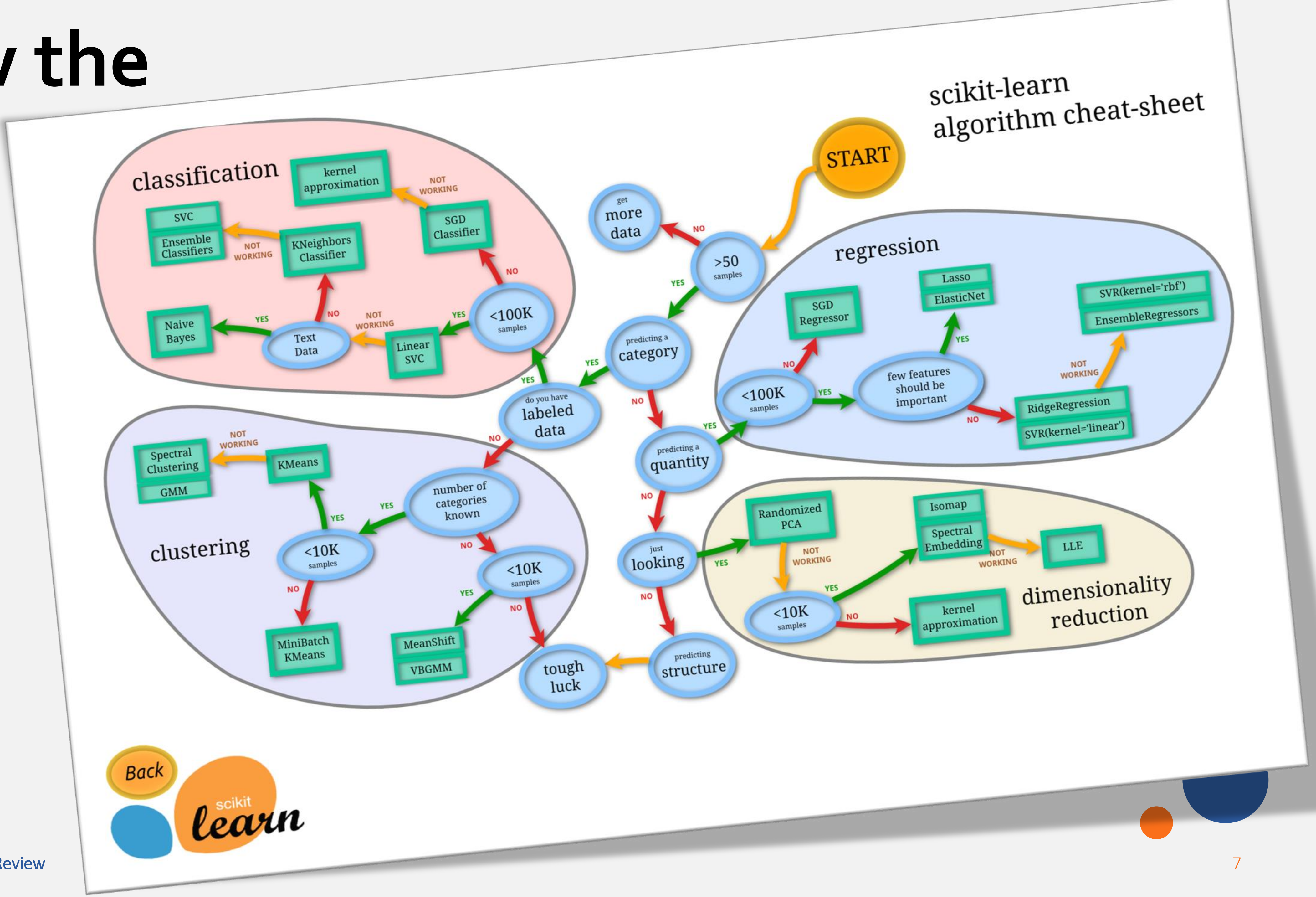
```
# Check if exist any missing values or categorical data .
housing_df.isna().sum()
```

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
target        0
dtype: int64
```

Choose the right algorithm!

How can I choose the right estimator for my case !!!!

Follow the map!



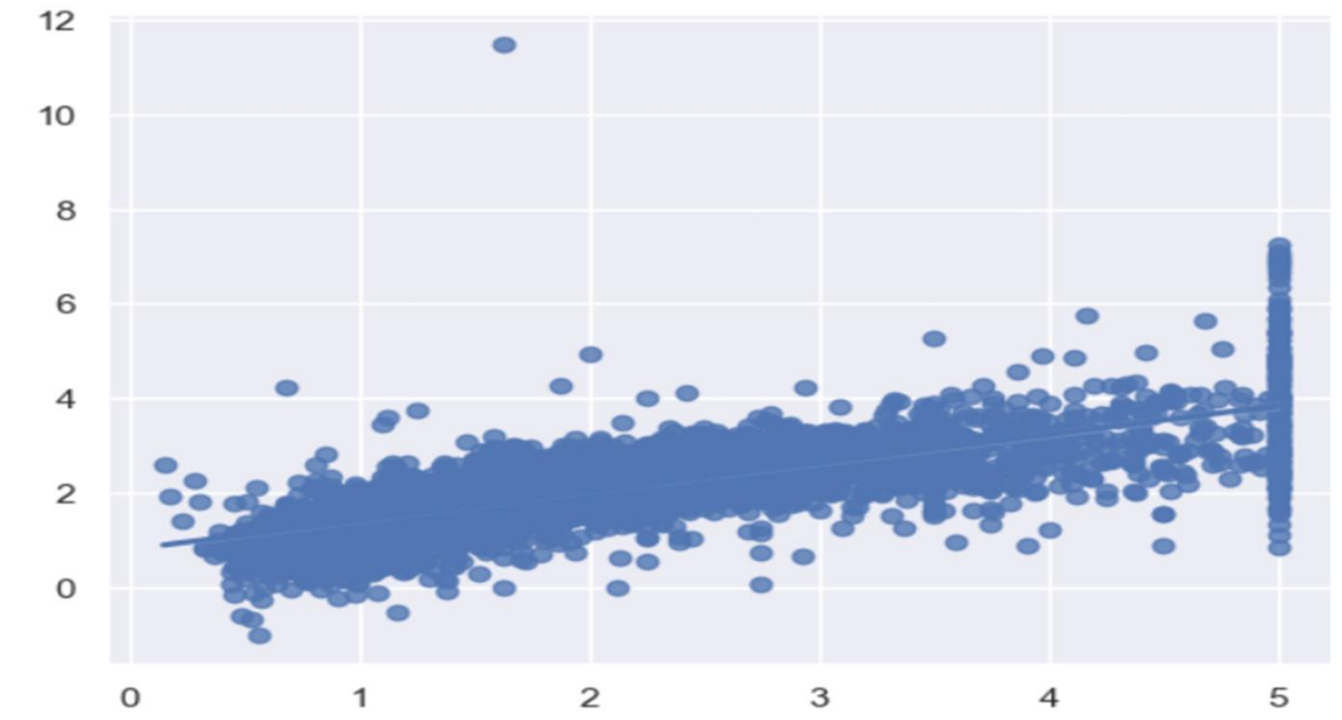
RidgeRegression estimator!

```
# Create x features , y label (target variable)
x= housing_df.drop("target",axis=1)
y = housing["target"]

# Split our data into training sets and testing sets
from sklearn.model_selection import train_test_split

x_train , x_test , y_train , y_test = train_test_split(x,y,test_size=0.2,random_state=42)

# choose the right estimator !!
# Experimetnal !!
from sklearn.linear_model import Ridge
model = Ridge()

# instantiate and fit the model on training sets
model.fit(x_train,y_train)

# Check the model score !!
model.score(x_test,y_test)
```

```
0.5758549611440126
```

RidgeRegression Plot.

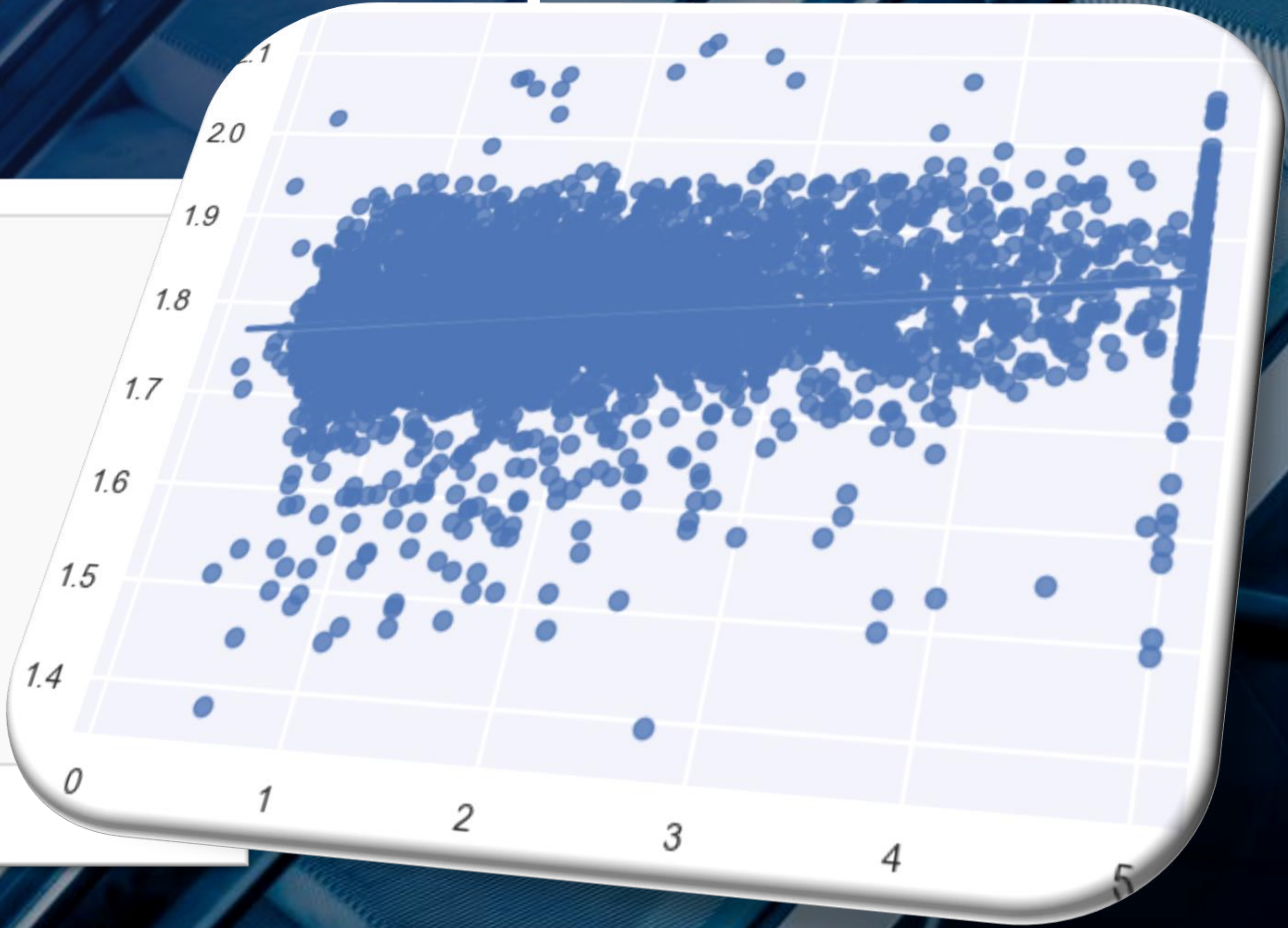
```
# Make predictions
y_preds = model.predict(x_test)
sns.regplot(x=y_test,y=y_preds);
```



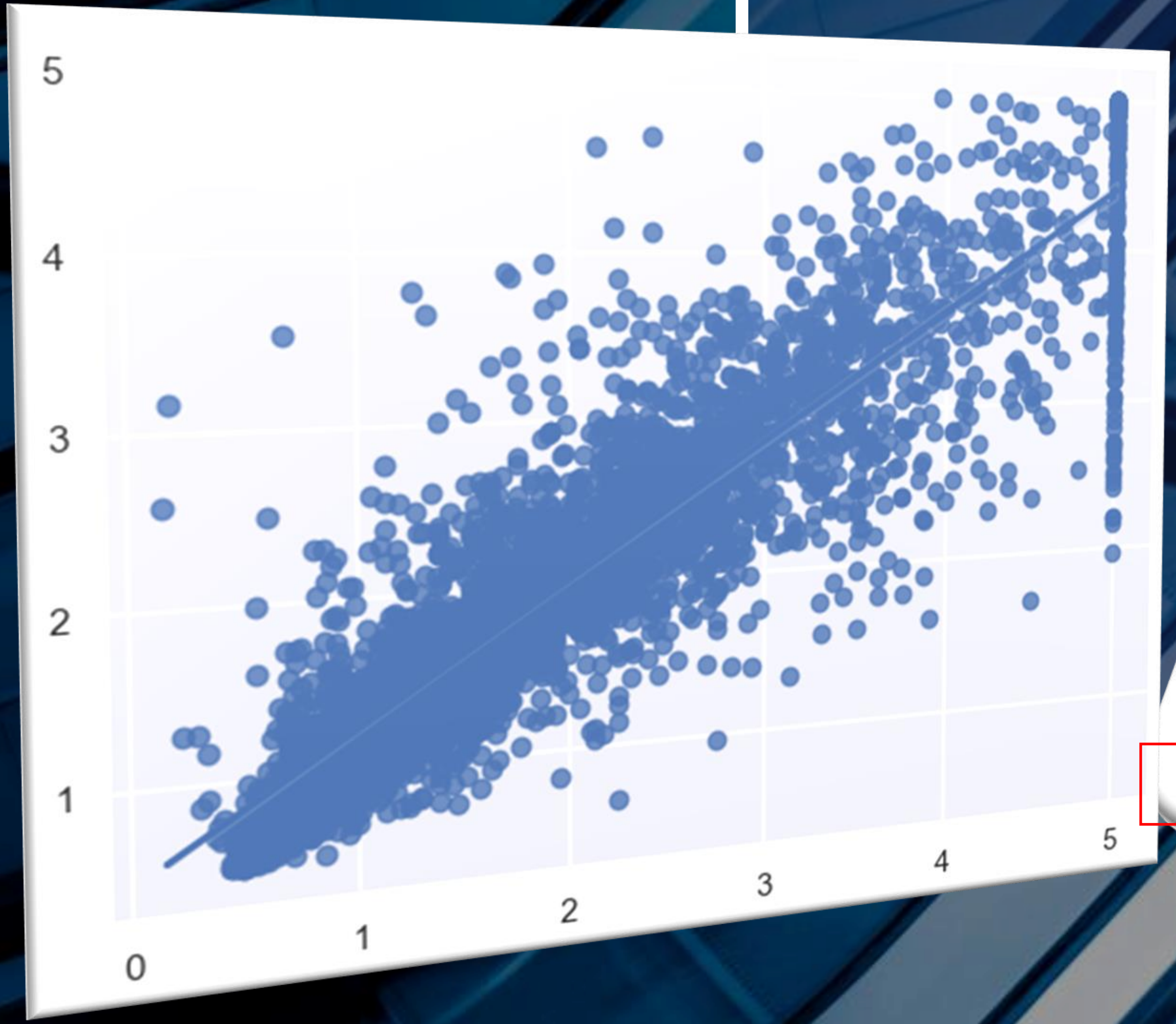
SVR estimator!

```python
# Choose another estimator !!
from sklearn.svm import SVR

# instantiate and fit the model on training sets
model = SVR()

model.fit(x_train,y_train)

# Check the model score on testing sets
model.score(x_test,y_test)
```

-0.01648536010717372



RFR estimator!




```python
# Choose another estimator (back to sciket learn map)
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()

# instantiate and fit the model on training sets
model.fit(x_train,y_train)

# Check the score of the model on testing sets
model.score(x_test,y_test)
```

```
0.8078513724240605
```

Evaluation Metrics

M1

Mean Absolute error (MAE)

the average of the absolute differences between predictions and actual values.

M2

Mean squared error (MSE)

is the mean of the square of the errors between actual and predicted values.

M3

Root Mean squared error (RMSE)

Is the Root of MSE

M4

R2 Score (R Squared)

How much variation of a dependent variable is explained by the independent variables

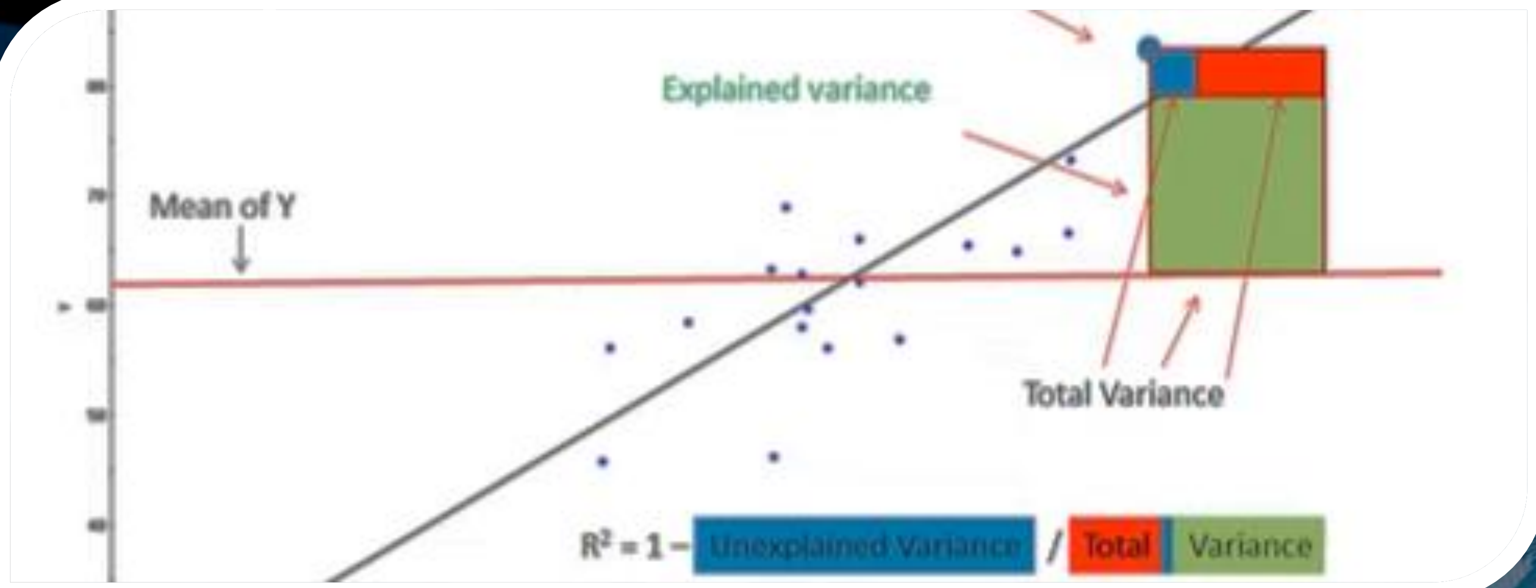
1.MAE (Mean Absolute Error)

(Mean Absolute Error "**MAE**")

- **Mean Absolute Error (MAE)** is the mean of the absolute value of the errors.
- **MAE** is the easiest to understand, because it's **the average error**.
- If MAE is **zero**, this indicates that the model predictions are perfect.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

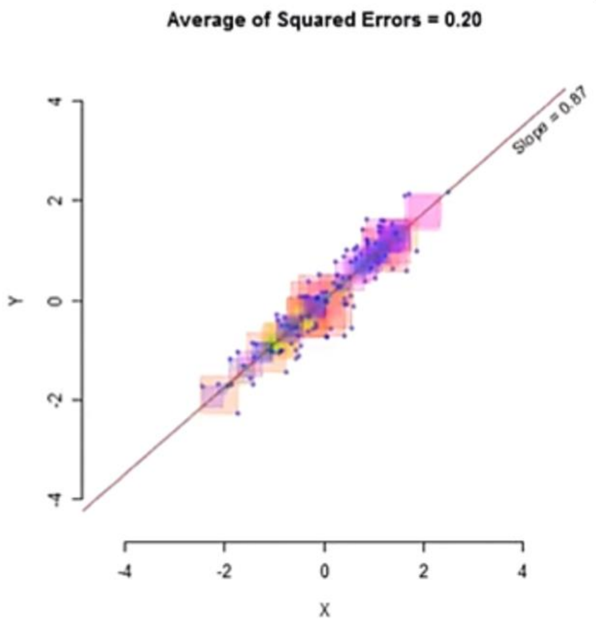
1.MAE (Mean Absolute Error)



MSE (Mean Squared Error)

- **Mean Squared Error** (MSE) is the mean of the squared errors.
- **MSE** values are generally **larger** compared to the MAE since the residuals are being squared.
- In case of data **outliers**, **MSE** will become much **larger** compared to MAE.
- **MSE** is more popular than MAE, because MSE "**punishes**" larger errors, which tends to be useful in the real world.

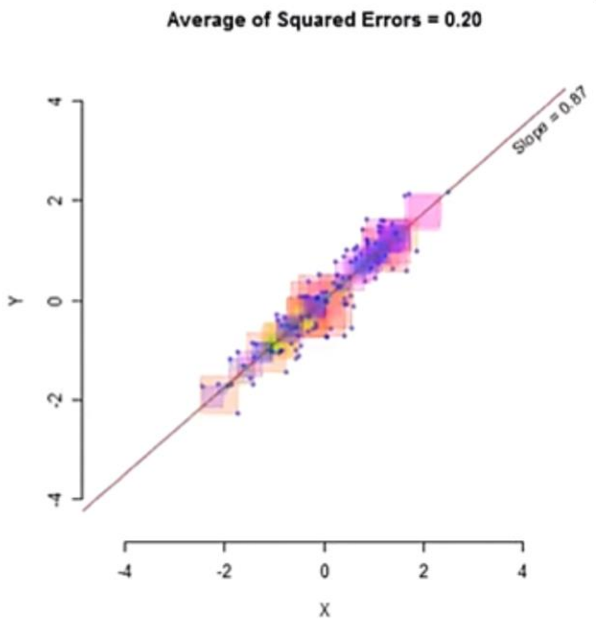
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$



MSE (Mean Squared Error)

- **Mean Squared Error** (MSE) is the mean of the squared errors.
- **MSE** values are generally **larger** compared to the MAE since the residuals are being squared.
- In case of data **outliers**, **MSE** will become much **larger** compared to MAE.
- **MSE** is more popular than MAE, because MSE "**punishes**" larger errors, which tends to be useful in the real world.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$



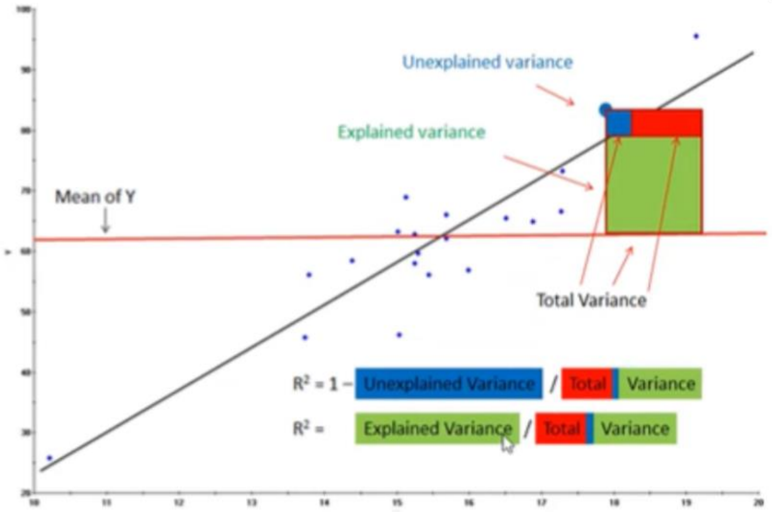
RMSE

The root of MSE

Let’s dive in

R2 (R Squared)

- **R-Squared** is a statistical measure of fit that indicates **how much variation** of a **dependent** variable is **explained** by the **independent** variable(s) in a regression model.
- If **R^2=70**, this means that **70%** of the **increase** in the **dependent** variable is due to **increase** in the **independent** variable.
- **R^2** provides an indication **of goodness of fit**.
- **R^2** value is a range between **(0)** “worst” and **(1)** “best”.



$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Improve a model!

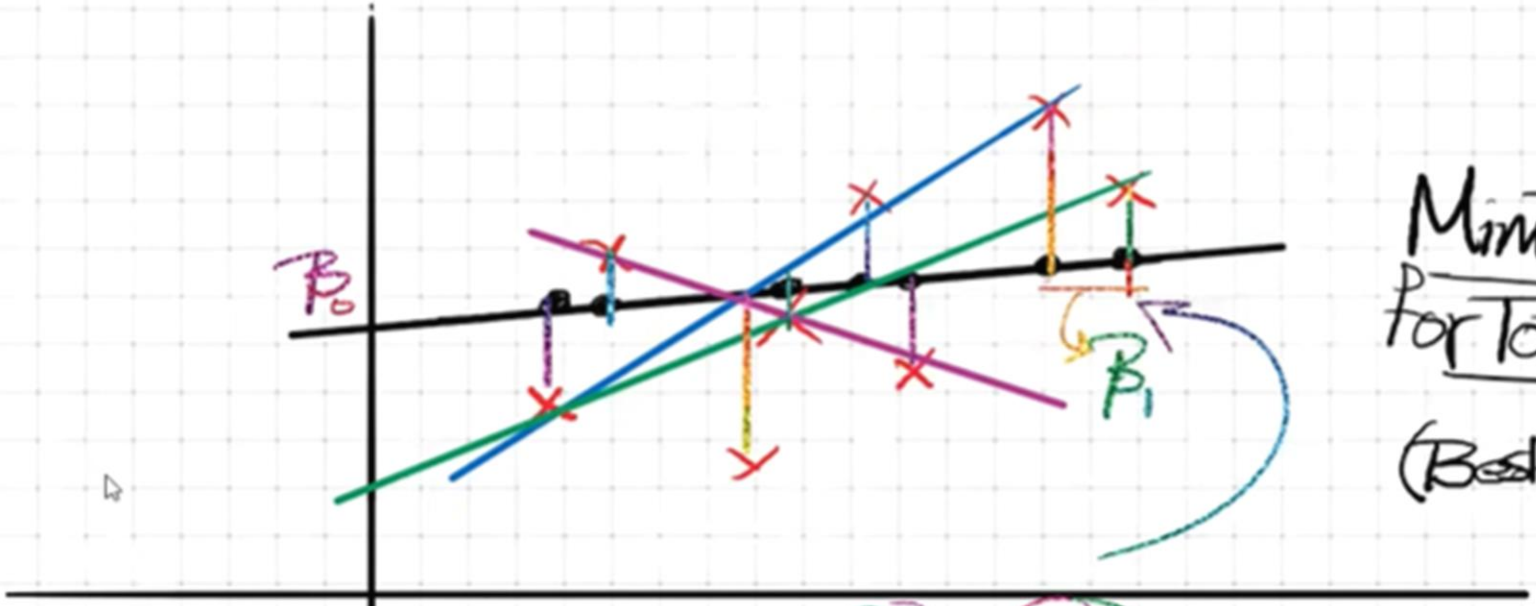
```python
# Improving our model !
# Try a different amount of n-estimators
for i in range(10,110,10):
    print(f"Trying model with {i} estimators...")
    model = RandomForestRegressor(n_estimators=i).fit(x_train, y_train)
    print(f"Model accuracy on test set: {model.score(x_test, y_test) * 100:.2f}%")
    print("")
```

```
Trying model with 10 estimators...
Model accuracy on test set: 78.02%

Trying model with 20 estimators...
Model accuracy on test set: 79.69%

Trying model with 30 estimators...
Model accuracy on test set: 80.42%

Trying model with 40 estimators...
Model accuracy on test set: 80.11%

Trying model with 50 estimators...
Model accuracy on test set: 80.47%

Trying model with 60 estimators...
Model accuracy on test set: 80.39%

Trying model with 70 estimators...
Model accuracy on test set: 80.62%

Trying model with 80 estimators...
Model accuracy on test set: 80.40%

Trying model with 90 estimators...
Model accuracy on test set: 80.74%

Trying model with 100 estimators...
Model accuracy on test set: 80.86%
```

$\Delta x \quad x_2 - x_1 \quad 10-5 \quad 5$

Model < Math. $\quad y = mx + b$

Stat. $\quad y = \beta_1 x + \beta_0 + \varepsilon$



Minimizing For Total Error

(Best-Fitted Line)

$$\hat{y} = \hat{\beta}_1 x + \hat{\beta}_0$$

Coefficients

THANK YOU!

Don't hesitate to get in touch with us !

I'm waiting for hearing from you

Khaled Shaker

Khaled Shaker
khaledgama4@gmail.com