PART - 2 [Regression]

- Regression models (linear & non linear) are used for predicting a real value (like salary).

  - If indep. var. is time, we are forecasting future values otherwise we are predicting present but unknown values.

Regressions:

1. Simple Linear R.
2. Multiple Linear R.
3. Polynomial R.
4. Support vector R. (SVR)
5. Decision Tree R.
6. Random Forest R.

1. Simple Linear R.

slope-coefficient

Dependent variable → $\hat{y} = b_0 + b_1 x_1$ ← indep. variable / predictor

y-intercept (constant).

DATE:
PAGE NO:

- y_i is the expected output
- $\hat{y}_i$ is the calculated output

There ~~are~~ is a difference all or most of time. Nothing ~~eeth~~ can be perfect.

$$\text{residual} = y_i - \hat{y}_i$$

$$\hat{y} = b_0 + b_1 x_1$$

Best eqn is the one such that (b_1 & b_0 are such that)
sum $(y_i - \hat{y}_i)^2$ is minimized.

② Multiple Linear R.

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \text{----} + b_n x_n$$

- Not all features are necessary. Some have more impact that we - decide by P-value of each feature.

5 methods of building models:
1. All in
2. Backward elimination
3. Forward selection
4. Bi-Directional elimination
5. All possible Models.

- We will be using backward elimination here in all models as it is the fastest.

- In multiple linear regression there is no need to apply feature scaling. Since the coefficients (weights) are multiplied by features. They get compensated.

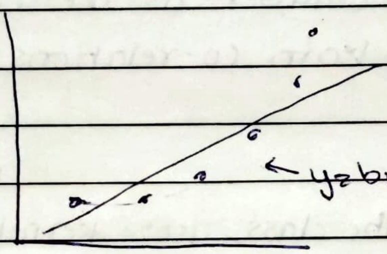
- Used with same LinearRegression library & identifies automatically as multiple Regression.

③ Polynomial Regression. (Linear)

$y = b_0 + b_1 x_1 + b_2 x_1^2 + \dots + b_n x_1^n$

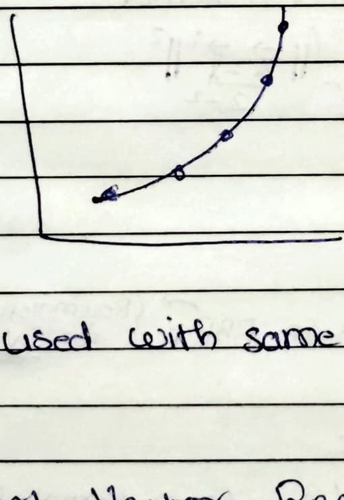
↓

Same variable with powers.



← Not a good fit.

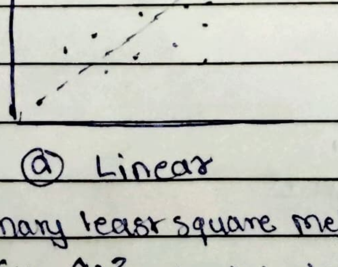
← $y = b_1 x_1 + b_0$.



- Since target is increasing exponentially it'll be better to use Poly. Regression

- Again used with same class.

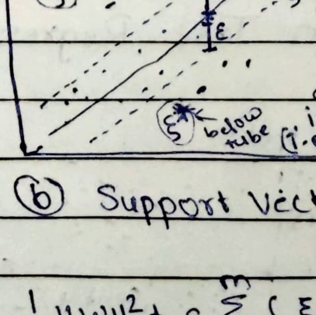
④ Support Vector Regression.



ⓐ Linear

ordinary least square method.

$SUM\ (y - \hat{y})^2 \rightarrow minimize$



ε = vertical distance (not perpendicular)

→ ε-insensitive tube
↓
disregarding errors inside this tube (i.e. margin of error)

ⓑ Support Vector.

$\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \rightarrow min$

DATE:
PAGE NO:

- Slightly more advanced since we would have to play a lot with feature scaling.

  we learn - (1) feature scaling transformation

  (2) Inverse transformation (to go back to original scaling).

- Need to apply feature scaling since we dont have coefficients multiplying to features in SVR.

- NO splitting in train data set as we want to leverage maximum ~~learning~~ data to learn co-relations in ~~res~~ dataset.

- During training SVR, the class uses kernels functions too ~~to~~ as args.

  One of the kernel, Gaussian RBF kernel

$$K(\vec{x}, \vec{l}^{\,i}) = e^{-\left(\frac{\|\vec{x} - \vec{l}^{\,i}\|^2}{2\sigma^2}\right)}$$

Some more kernel functions.

(1) Polynomial Kernel

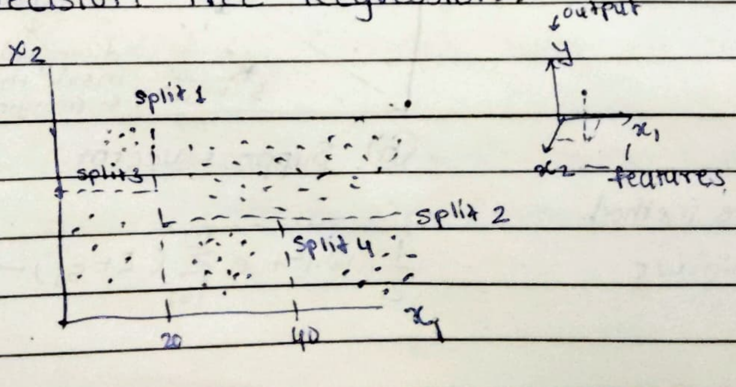
(2) Gaussian K. & Gaussian RBF K. (Recommended for SVR)

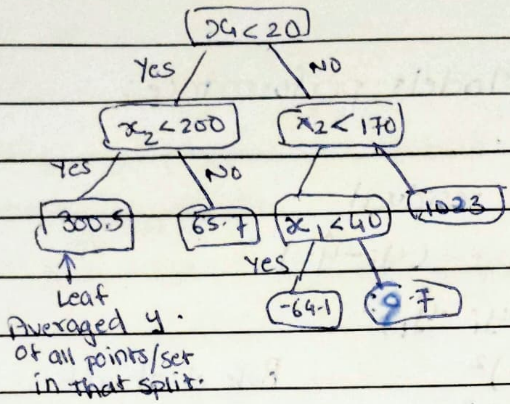
(3) Laplace RBF K.

(4) Sigmoid K. e.t.c.

More on data-flair.training/blogs/svm-kernel-functions/

(4) Decision Tree Regression.





- The decision Tree works on information gain & Entropy (have mathematical formulas) & decide split where information gain is maximum.

- As for the regression task, the averaged output is taken from tree at leaf. (averaged o/p of all datapts inside split)

- We dont need to scale values here, since predictions are resulting from random splits of data & not some eq^ns. So there is no problem of a feature being too overwhelming with large values.

- Recommended only for High dimension dataset (more features).

(5) Random Forest Regression (Also recommended mostly for high dimension data i.e. more feature set.)

- Random forest is one of method of ensemble learning. One other method is gradient boosting.

- Ensemble learning is when we take multiple algos. or same algo. multiple times & put it together to make better model than the single original model.

STEPS: (1) Pick K random data points from training set.
(2) Build Decision Tree associated with these K data pts.
(3) Choose N trees to build & repeat step 1 & 2.
(4) For predict, Predict Y value from N models & take avg. of all Y.

DATE:
PAGE NO:

Evaluating Regression Models performance:

- (USING R SQUARED).

SS_{res} ⇒ sum of squared residual

residual / loss = $(y_i - \hat{y}_i)$

sq. loss = $(y_i - \hat{y}_i)^2$

$SS_{res} = SUM(y_i - \hat{y}_i)^2$

$SS_{tot} = SUM(y_i - y_{avg})^2$ from train set data points.

$$\boxed{R^2 = 1 - \frac{SS_{res}}{SS_{tot}}}$$

- lesser the SS_{res} the better the model is.

Rule of thumb:

R^2 = 1.0 = Perfect fit (suspicious)

~ 0.9 = very good

< 0.7 = Not great

< 0.4 = Terrible

< 0 = Model makes no sense for given data.

- (Adjusted R Squared).

Problem:

$\hat{y} = b_0 + b_1x_1 + b_2x_2$ (+ b_3x_3)

we add a 3rd feature or more extra features.

- The SS_{tot} doesnt change ($SS_{TOT} = SUM(y_i - y_{avg})^2$)
- but SS_{res} changes & will decrease or stay same. & never increase.

∴ [This is coz of ordinary least sq. method : $SS_{res} \rightarrow$ (min)]

$$Adj\ R^2 = 1 - (1 - R^2) \times \frac{n-1}{n-k-1}$$

- k = no. of independent variables
- n = sample size.

look for k.
- if k increases, ratio increases & adj R decreases (substraction)

- For Selecting Best Model, simply train all model and compare their r^2 scores.

DATE
PAGE NO.

SUMMARY

① Simple Regression
- Simple regression on single feature minimizing SSres.

* IMPORT : from sklearn.linear_model import LinearRegression.

② Multiple Regression
- Multiple features
- May need to preprocess, for e.g. encode data

*IMPORT : from sklearn.linear_model import LinearRegression
- Cant plot graph since its multidimensional.

③ Polynomial Regression
- Output scales polynomially with input
- Need to decide degree of polynomial.

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2^2 + b_3 x_3^3 + \cdots b_n x_n^n$$

- If n too large it'll try to overfit through all pts.

* IMPORTS :
- for model we need to create more features of x, for each ~~feature of x~~

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg= PolynomialFeatures(degree =4)
x_poly= poly_reg.fit_transform(x)
from sklearn.linear_model import LinearRegression
```

④ Support Vector
- Need to rescale all features & dependent variable in SVR.

* IMPORT: from sklearn.svm import SVR

```
regr = SVR(kernel ='rbf').
```

- Uses Kernel functions
- Need transform scale to ~~from~~ train & inverse transform output scale for proper output.
- Rescale w.r.t their scaler for x & diff. scaler for y.

(5) Decision Tree.

- works on entropy & splits on highest.

* IMPORTS: from sklearn.tree import DecisionTreeRegressor.
  dt = DecisionTreeRegressor(random_state=0)

(6) Random Forest

- ensemble method

* IMPORTS: from sklearn.ensemble import RandomForestRegressor
  rf = RandomForestRegressor(n_estimators = 10, random_state=0)

Evaluating Performance:

- from sklearn.metrics import r2_score