E-F ⟷ SVM ⟷ Doubt ⟷ Job ⟷ SVM code
(4)

Date.....10.........

Encoding - Theory

- 1) Label encoding
- 2) One-Hot Encoding (pd.dummy) [0 or 1]
- 3) Ordinal Encoding

↳ when Cat feat is not ordinal, The no. of categorical feat is less

↳ when cat feat is ordinal, The no. of categorical feat is quite large.

① Label Encoding (each label/class is assigned a unique integer based on alphabetical ordering)

↓

→ $\boxed{\text{alp order NOT order of Data}}$

→ from sklearn.preprocessing import LabelEncoder

→ le = LabelEncoder()

→ y = le.fit_transform(y).

[Drawback → False order of Data]

② One-Hot Encoding [creates diff col for classes of a feat, and 1 → if it appears and 0 → if not]

Drawback → ↑ in Dimensionality

——— x ——— x ——— x ———

- Tree-Based algos → can work with [cat variables and Label Encoding]
- LR, Distance metric (K-m, KNN) or ANN
  ↳ works with OH

Spiral Encoding

Date.....................

= one-df = pd.get_dummies(y, drop_first = True)

→ df = pd.concat([df, one-df], axis=1)

→ df = df.drop([' '], axis=1).

3)• ordinal Encoding → as per order of Data [Label]

→ Cat feat → ordinal num value [ordered set].

→ eg →	poor	1	Not alphabetic order But order of Data.
	Good	2	
	Very Good	3	
	Excellent	4	

creating a dict for mapping (with values)

```
dt-dict = {'poor': 1, 'Good': 2, 'Very Good': 3,
 'Excellent': 4}

df['Column'] = df['Customer Rat'].map(df-dict)
```

---- x ---- x ---- x ----

Feature Scalling [After split on training Data]
[value range 0-1]

→ Normalization and Standardazation.

[min-max] [Z-score]

[value end up B/w 0 and 1] [centered around mean with value of std]

$\boxed{\frac{X - X_{min}}{X_{max} - X_{min}}}$.

$\boxed{\frac{X - \mu}{\sigma}}$. └ Kitna std deviation from mean

Spiral

fit → transform

Date..................

Normalize → when we know that Distribution of our data does not follow a Gaussian distribution [KNN and NN]

Standarlization → where Distribution of our data follows Gaussian distribution

filter the numeric

```
df_num = df.select_dtypes(include = np.number)
```

```
# normalization
from sklearn.preprocessing import minMaxScaler
 norm = minMaxScaler().fit(data_num)
 data_nom_norm = norm.transform([ ])
```

here this or feature you want to normalize.

```
# standardization
 from sklearn.preprocessing import StandardScaler
 scale = Standard Scaler(). fit(data_num)
 data_num_scale = scale.transform(data_num)
```

———— x ———— x ———— x ————

* Missing values

① Check assigned Data types

if wrong → change using astype()

② Standard miss value → detect By Python (NaN)

Date..................

Outliers → detect using Boxplot

```
df_num = df.select_dtypes(include = np.number)
fig, ax = plt.subplots([2,2], figsize=(15,8))

for var, subplot in zip(df.columns, ax.flatten()):
    z = sns.boxplot(x= df[var], orient='h', ax=subplot)
    z.set_xlabel(var, fontsize=20)
```

① To drop them

IQR method

Z-score (>, < -3)

② To treat Them

Log transformation

Quantile based flooring and capping

Measuring SLR, MLR, PR

Date.........

1) MSE value
2) R^2 value

SLR → Residual plot
MLR → dist plot
PR → poly func plot

↓ MSE val and ↑ R^2 val (R^2 → coeff of Determinant)

sum of squares due to Regression

$SST = SSR + SSE$

Total sum of squares

sum of squares due to error.

- $\boxed{r^2 = \frac{SSR}{SST}}$ or → $\frac{SST}{SST} = \frac{SSR}{SST} + \frac{SSE}{SST}$

$1 = r^2 + \frac{SSE}{SST} \rightarrow r^2 = 1 - \frac{SS_{reg}}{SS_{tot}}$

- Sample Correlation Coeff

$r_{xy} = (\text{sign of } b_1)\sqrt{\text{coeff of Determinant}}$

$r_{xy} = \boxed{(\text{sign of } b_1)\sqrt{r^2}}$

2nd way of calc $\boxed{R^2}$ →

$R^2 = 1 - \left[\frac{\text{MSE of reg line}}{\text{MSE of avg of data}}\right]$.

$\boxed{1 - \frac{\Sigma(y_i - \hat{y}_i)^2}{\Sigma(y_i - \bar{y})^2}}$

MSE

$\boxed{\frac{\Sigma(y - \hat{y})^2}{n}}$.

Spiral

Reg performance

Date.....................

→

- For SLR → MSE value + R^2 score

For MLR → MSE value + Adj R^2 score

Adj R^2 score ⇒ $1 - \left[\frac{(1-R^2)(N-1)}{N-p-1}\right]$

N = sample size

p = No. of predictors.

* Every time we add a new feat, R^2 ↑

- (even though the feat has low corr)

- But adj R^2 ↑ only when feat has high corr

$\boxed{\text{Adj } R^2 \leq R^2}$.

```
for var in df.unique
        i = 1
   df-dict = {var: i}
        i = i+1;
```