feature
1.continuous numeric
2.categorical feature

vector space = feature matrix = x
label = ground truth = y

Distance base (NN-centroid, KNN, KD tree)
fast, least memory, scalability | sensitive to outliner

Preprocess ทำให้ Data เรียบ
① null/nan
1.1 ลบ row
1.2 เติมค่า mean ← continuous
mode ← categorical
1.3 close fit ← nearest val
② outliner ไม่เหมาะ data ใหญ่
2.1 plot & erase (visually)
ค้นผลและอะไร
2.2 คำนวณ SD แล้วลบพวกไกล จาก mean (statistically)

$$SD = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n-1}}$$

③ Data smooting
3.1 smooting
sort → split → replace { boundary, mean, median }
↑ bin
3.2 Feature scaling
Decimal scaling $v' = \frac{v}{10^j}$; v: arr[i], j = min val

minmax normalization
$$v' = \frac{v - min_{เก่า}}{max_{เก่า} - min_{เก่า}} (max_{ใหม่} - min_{ใหม่}) + min_{ใหม่}$$

z-score $v' = \frac{v - mean_{เก่า}}{SD_{เก่า}}$

④ Feature selection
4.1 Feature selection ลบ column ไม่ใช้
4.2 Feature aggregation รวม/ผสม
ไอเดีย vector

corr() $r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$ ← continuous

Mutual information ← เยอะ = สัมพันธ์

		outlook			
		sunny	overcast	rain	
temp	hot	2	2	0	4
	cold	1	1	2	4
	mild	2	1	3	6
		5	4	5	14

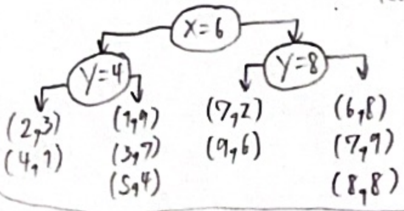
$$P(outlook = sunny, temp = hot) = \frac{2}{14} \cdot \log_2 \left(\frac{2/14}{\frac{5}{14} \times \frac{4}{14}} \right)$$

⑤ class imbalance

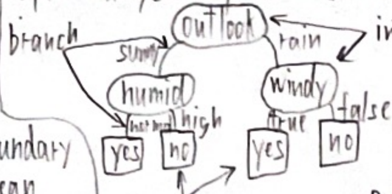
NN-centroid (mean vector)
1.หา mean แยก label (แยก feature)
2.คำนวณเทียบกับละ label ด้วย
$\sqrt{\sum (avg\ col_i - sample\ col_i)^2}$ code หา √
3.น้อยสุด = predict
hyperparameter: จำนวน k →

KNN
1.คำนวณกับ ทุก row
$\sqrt{(x_{ใหม่} - x_i)^2 + (y_{ใหม่} - y_i)^2}$
2.เลือกน้อยสุด k ตัว
1-NN: ไม่ทน outliner
KNN: ทนกว่า แต่อาจมีมากกว่า 1 คำตอบ

KD tree
1.sort data
2.หา median มา if else แบ่ง data
3.if len >= leaf size: loop
↓ else
4. KNN ปกติ
hyper: k, leaf size



D tree (hierachical structor)
explainable, ez implement | brute force หา decision

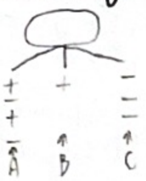


เยอะ = ดี เพราะมัน แบ่งได้ entropy ลดเยอะ

Entropy = วัดความไม่แน่นอน ← 0 = ดีสุด 1 = แย่สุด
$$H(S_1) = -\left[\frac{2}{5} \log_2 (2/5) + \frac{3}{5} \log_2 (3/5) \right]$$

Gini Index = ค่าความน่าจะเป็นที่จะเดาผิด
$$Gini(S_1) = 1 - \left[(2/5)^2 + (3/5)^2 \right]$$ ← ยิ่งน้อยยิ่งดี

information gain



entropy เก่า
$$IG(-,-) = 0.953 - \left[\frac{4}{8} H(A) + \frac{1}{8} H(B) + \frac{3}{8} H(C) \right]$$

จะแยกจนกว่าจะ:
① entropy ทุกกิ่ง = 0
② information ไม่เพิ่ม

Bagging (Random forest)
+) แม่นยำ + explainable
-) data ต่ออัน น้อย

Boosting (XGBoost)
สร้าง weak (depthน้อย) มาแก้ข้อผิด

point	y_i	$\hat{y}_i$	sigmoid	err (y_i - sigmoid)

← epoch แยกส่วน $\hat{y}_i$
หลังๆคำนวณโดย $\hat{y}_1 = \hat{y}_{0i} + \alpha \cdot hv$ ← avg err
คำนวณแยกครั้งโดยใช้ lastest decision boundary
sum(err)/n ของการ ครั้งนั้น

linear model

x หาร w_0 → $\hat{y}_i - y_i$

$2\sum(\hat{y}_i - y_i) \cdot x_i$ → G(w1)

$2\sum(\hat{y}_i - y_i)$ → G(w0)

$\sum(err)$ → Loss

LR	x	y	w1	w0	ŷ	err	G(w1)	G(w0)	Loss	Epoch
0.2	0.1	0.5	0.5	0.5	0.55	0.05	0.700	1.00	0.165	1
	0.5	0.7			0.75	0.05				
	0.8	0.5			0.9	0.40				
	0.1	0.5	0.36	0.3	0.34	-0.164	-0.112	-0.592	0.083	2
	0.5	0.7			0.48	-0.22				
	0.8	0.5			0.59	0.088				
	0.1	0.5	0.382	0.418	0.46	-0.043	0.26	0.181	0.060	3
	0.5	0.7			0.61	-0.090				
	0.8	0.5			0.72	0.224				

$$x_{i+1} = x_i - \alpha \cdot \nabla f(x_i)$$ ← $G(w_i)$

```
import pandas as pd
df = pd.read_csv('file name')
df = df.loc[(df['a']>0) & (df['b']<= 10)]
df = df[['col 1','col 2']]
df = df.drop(columns=['col 1','col 2'])
df.isna().sum()     df = df.dropna()
df = df.fillna(0) / df.fillna(df.mean(numeric_only=True)) / df.fillna(df.median()) ← continuous
df = df.fillna(df.select_dtype(include='object').mode().iloc[0]) ← categorical
df['col...'] = df['col...'].replace({'ค่าเก่า':ค่าใหม่, 'ค่าเก่า':ค่าใหม่})
df = pd.DataFrame(arr, columns=['col 1','col 2']) ← arr to df
cor = df.corr() / df.corr()['col ~'] | df = df[cor.index[:5]]
df = df.sort_values(by='col ~', ascending=False) ← sort ก่อน มากไปน้อย
```

Logistic model

$w_{i+1} = w_i - \alpha G(w_i)$ $\quad \frac{1}{1+e^{-z}}$ $\quad \Sigma(O_i - Y_i)A$ $\quad \Sigma(O_i - Y_i)B$ $\quad \Sigma(O_i - Y_i)$ $\quad \Sigma(err)$

A	B	Y	W1	W2	W0	Z	sigmoid	err	G(W1)	G(W2)	G(W0)	BCE loss	epoch
0	0	0	0.3	0.3	0.3	0.3	0.57	1.23	0.36	0.36	1.58	4.72	1
0	1	0				0.6	0.65	1.50					
1	0	0				0.6	0.65	1.50					
1	1	1				0.9	0.71	0.49					
0	0	0	0.23	0.23	-0.02	-0.02	0.5	0.99	0.16	0.16	1.21	4.03	2
0	1	0				0.21	0.55	1.16					
1	0	0				0.21	0.55	1.16					
1	1	1				0.44	0.61	0.72					

$err: -[Y\log_2(\hat{Y}) + (1-Y)\log_2(1-\hat{Y})]$

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder
scaler = MinMaxScaler(feature_range=(-1,1)) ← ช่วง dis
scaler = StandardScaler() ← ช่วง math | df = pd.DataFrame(scaler.fit_tranform(df), columns=df.columns)
labEn = LabelEncoder() | df['col-'] = labEn.fit_tranform(df['col~'])
```

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figue(figsize=(8,6))
sns.heatmap(df.corr(), annot=true)
plt.title('Title name')
plt.shown()
```

```
import pickle
pickle.dump(model, open('~.sav/.pkl', 'wb'))  ← save model
top.to_csv('~.csv', index=False) ← save.csv
```

```
y_pred = model.predict(xtest)
```

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred
print(r2) ← continuous
```

```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, model.predict(xtest)) ← categorical
```

```
df['col'].shift(1) ← ลง
df['col'].shift(-1) ← ขึ้น
```

```
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size=0.2, random_state=42)
```

str to datetime

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True) ← ใช้ Date แทน index
df['Day'] = df['Date'].dt.day
df['Week'] = df['Date'].dt.isocalendar().week
df['Mount'] = df['Date'].dt.month
df['Year'] = df['Date'].dt.year
```

```
from sklearn.model_selection import GridSearchCV
parameters = {'n_neightbor': [1,20]} ← range
model = GridSearchCV(model, parameters)
model.best_params_
model.best_score_   } หลัง fit
```

```
from sklearn.neighbors import KNeighborsClassifier, NearestCentroid
model = KNeighborsClassifier(n_nighbors=5) ← dis, ind = knn.KNeighbors([xtest], n_neighbors=5) | print(dis, ind)
model = NearestCentroid()
```

```
import xgboost as xgb
model = xgb.XGBClassifier(objective="multi:softmax",
    n_estimators=10, max_depth=5, num_class=2)
```

```
from sklean.linear_model import LinearRegression
model = LinearRegession()
model = fit(xtrain, ytrain)
print('w1:', model.coef_)
print('w0:', model.intercept_)
```

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(max_depth=2, criterion='entropy')
a = clf.best_estimator_
for i inrange(6):
    plt.figure(figsize=(20,10))
    plot_tree(a.estimators_[i], feature_names=xTrain.columns,
    plt.title(f'tree{i+1}')
    plt.show()
class_name = ['<=50k', '>50k'])
```

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
model = DecisionTreeClassifier(max_depth=2, criterion='entropy')
import matplotlib.pyplot as plt
plt.figure(figsize=(20,10))
plot_tree(model, feature_name=xtrain.columns, class_names=
plt.show()
label ——> ['<=10k', '>10k'])
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
print("MAE: ", mean_absolute_error(ytest, y_pred)) ← [illegible]
print("MSE: ", mean_squared_error(ytest, y_pred)) ← outliner
```

นายภัทรศักดิ์ พลอยสัมฤทธิ์
6604062610225