소프트웨어학과 32204041 정다훈 7장 과제

In [1]:

```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

# Load the data
df = pd.read_csv('C:\dankook\DeepLearning_Cloud\data\madelon.csv')
print(df.head())

df_X = df.loc[:, df.columns != 'class']
df_y = df['class']

# df_y의 값이 1, -1로 되어 있어서 1, 0으로 변경
df_y = df_y.replace(1, 1)
df_y = df_y.replace(-1, 0)
```

```
   class   V2   V3   V4   V5   V6   V7   V8   V9  V10  ...  V492  V493  V494  \
0     -1  485  477  537  479  452  471  491  476  475  ...   477   481   477   
1     -1  483  458  460  487  587  475  526  479  485  ...   463   478   487   
2     -1  487  542  499  468  448  471  442  478  480  ...   487   481   492   
3      1  480  491  510  485  495  472  417  474  502  ...   491   480   474   
4      1  484  502  528  489  466  481  402  478  487  ...   488   479   452   

   V495  V496  V497  V498  V499  V500  V501  
0   485   511   485   481   479   475   496  
1   338   513   486   483   492   510   517  
2   650   506   501   480   489   499   498  
3   572   454   469   475   482   494   461  
4   435   486   508   481   504   495   511  

[5 rows x 501 columns]
```

In [2]:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from xgboost import XGBClassifier
from sklearn.feature_selection import SelectKBest, chi2

# Feature selection using the filter method
test = SelectKBest(score_func=chi2, k=df_X.shape[1])
fit = test.fit(df_X, df_y)

# Sort features by their scores
f_order = np.argsort(-fit.scores_)  # sort index by decreasing order
sorted_columns = df.columns[f_order]

# Test classification accuracy by selected features using XGBoost
model = XGBClassifier(eval_metric='logloss', random_state=1234)

df_X_best = []
temp = 0
for i in [30,50,70]:
    fs = sorted_columns[0:i]
    df_X_selected = df_X[fs]
    scores = cross_val_score(model, df_X_selected, df_y, cv=5)
    print(fs.tolist())
    print(np.round(scores.mean(), 4))
    if temp < scores.mean():
        temp = scores.mean()
        df_X_best = df_X_selected
```

```
['V106', 'V476', 'V337', 'V65', 'V494', 'V339', 'V242', 'V443', 'V454', 'V379',
'V49', 'V473', 'V154', 'V412', 'V137', 'V434', 'V330', 'V205', 'V212', 'V348', 'V
11', 'V57', 'V150', 'V282', 'V5', 'V495', 'V432', 'V287', 'V129', 'V200']
0.5125
['V106', 'V476', 'V337', 'V65', 'V494', 'V339', 'V242', 'V443', 'V454', 'V379',
'V49', 'V473', 'V154', 'V412', 'V137', 'V434', 'V330', 'V205', 'V212', 'V348', 'V
11', 'V57', 'V150', 'V282', 'V5', 'V495', 'V432', 'V287', 'V129', 'V200', 'V47',
'V176', 'V297', 'V222', 'V482', 'V74', 'V86', 'V459', 'V247', 'V25', 'V120', 'V33
4', 'V286', 'V246', 'V413', 'V42', 'V194', 'V383', 'V415', 'V225']
0.509
['V106', 'V476', 'V337', 'V65', 'V494', 'V339', 'V242', 'V443', 'V454', 'V379',
'V49', 'V473', 'V154', 'V412', 'V137', 'V434', 'V330', 'V205', 'V212', 'V348', 'V
11', 'V57', 'V150', 'V282', 'V5', 'V495', 'V432', 'V287', 'V129', 'V200', 'V47',
'V176', 'V297', 'V222', 'V482', 'V74', 'V86', 'V459', 'V247', 'V25', 'V120', 'V33
4', 'V286', 'V246', 'V413', 'V42', 'V194', 'V383', 'V415', 'V225', 'V431', 'V29
9', 'V165', 'V13', 'V463', 'V56', 'V112', 'V115', 'V295', 'V464', 'V420', 'V256',
'V278', 'V344', 'V45', 'V353', 'V458', 'V288', 'V378', 'V418']
0.5175
```

In [3]:

```
######################################################################
# Forward Search
######################################################################
from sklearn.feature_selection import SequentialFeatureSelector
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score

# Define the model
model = XGBClassifier(eval_metric='logloss', random_state=1234)

# Initialize Sequential Feature Selector
sfs = SequentialFeatureSelector(model, direction='forward', n_features_to_select

# Fit the SFS model
fit = sfs.fit(df_X_best, df_y)
print("Num Features: %d" % fit.n_features_in_)
fs = df_X_best.columns[fit.support_].tolist()   # selected features
print("Selected Features: %s" % fs)

# Evaluate model performance with selected features
scores = cross_val_score(model, df_X[fs], df_y, cv=5)
print("Acc: " + str(scores.mean()))
```

```
Num Features: 70
Selected Features: ['V473', 'V495', 'V47', 'V222', 'V42']
Acc: 0.5635
```

In [4]:

```
######################################################################
# Backward elimination (Recursive Feature Elimination)
######################################################################
from sklearn.feature_selection import RFE
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score

# Define the model
model = XGBClassifier(eval_metric='logloss', random_state=1234)
```

```
# Initialize Recursive Feature Elimination
rfe = RFE(model, n_features_to_select=5)

# Fit the RFE model
fit = rfe.fit(df_X_best, df_y)
print("Num Features: %d" % fit.n_features_)
fs = df_X_best.columns[fit.support_].tolist()   # selected features
print("Selected Features: %s" % fs)
#print("Feature Ranking: %s" % fit.ranking_)

# Evaluate model performance with selected features
scores = cross_val_score(model, df_X[fs], df_y, cv=5)
print("Acc: " + str(scores.mean()))
```

```
Num Features: 5
Selected Features: ['V330', 'V495', 'V459', 'V286', 'V45']
Acc: 0.5465000000000001
```

Model_stacking

In [5]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import StackingClassifier
from catboost import CatBoostClassifier
from sklearn.svm import SVC
#standard scaler import

from sklearn.preprocessing import StandardScaler
import numpy as np
```

In [6]:

```
# Base model: XGBoost
base_model = XGBClassifier(eval_metric='logloss', random_state=1234)

# Cross-validation score for the base model
base_score = cross_val_score(base_model, df_X_best, df_y, cv=5)
print(f'score1: {np.mean(base_score)}')
```

```
score1: 0.5174999999999998
```

In [7]:

```
# Define the estimators for the stacking classifier
estimators = [
    ('rf', RandomForestClassifier(n_estimators=10, random_state=1234)),
    ('svm', SVC(probability=True, random_state=1234)),  # Support Vector Machine
    ('catboost', CatBoostClassifier(random_state=1234, verbose=0)),  # CatBoost
    ('lr', make_pipeline(StandardScaler(), LogisticRegression(max_iter=5000)))
]

# Define the stacking model
model_stacking = StackingClassifier(
    estimators=estimators,
    final_estimator=LogisticRegression(max_iter=1000)  # Using Logistic Regressi
)

# Cross-validation score for the stacking model
model_stacking_score = cross_val_score(model_stacking, df_X_best, df_y, cv=5)
print(f'score2: {np.mean(model_stacking_score)}')
```

```
score2: 0.5155
```