Mini Assignment 5. Feature Selection

Please add one code cell after each question and create a program to answer the question. Make sure your code runs without error. After you are finished, click on File -> Print -> Save as PDF to create a PDF output and upload it on Brightspace.

NOTE: Make sure your PDF does not have your name or any identifying information in the name or content of the file. Anonymity is essential for the peer-review process.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
```

The Case: Santander Customer Satisfaction

You are provided with an anonymized dataset on [Santander Bank]() customers. It contains a large number of numeric variables. You may [download the dataset here](). The "TARGET" column is the variable to predict. It equals one for unsatisfied customers and 0 for satisfied customers.

The task is to predict the whether that each customer in the test set is an unsatisfied customer.

1. Read the dataset into a dataframe, and check out the first few rows and column data types. What is the shape of the dataset (number of rows and columns)?

```
df = pd.read_csv('/content/drive/MyDrive/Class Notes/colabData/SantanderBank.csv')
df.dtypes
```

```
ID                           int64
var3                         int64
var15                        int64
imp_ent_var16_ult1         float64
imp_op_var39_comer_ult1    float64
                            ...
saldo_medio_var44_hace3    float64
saldo_medio_var44_ult1     float64
saldo_medio_var44_ult3     float64
var38                      float64
TARGET                       int64
Length: 371, dtype: object
```

```
df.isna().sum()
```

```
ID                         0
var3                       0
var15                      0
imp_ent_var16_ult1         0
imp_op_var39_comer_ult1    0
                          ..
saldo_medio_var44_hace3    0
saldo_medio_var44_ult1     0
saldo_medio_var44_ult3     0
var38                      0
TARGET                     0
Length: 371, dtype: int64
```

```
df.head()
```

2. Create predictor (X) and target (y) datasets.

```
X_raw = df.drop(["TARGET", "ID"], axis=1)
y = df.TARGET
```

Selection Using Feature Information

3. Create a mask that identifies predictor features with 5% and more variance. Make sure to normalize the predictors before you do this, because variance is highly sensitive to feature scales.

Use the mask to drop the features of low variance, and assigned it back to X. Display the shape of the resulting X.

```
X_norm = X_raw/X_raw.mean()

var_mask = X_norm.var() > 0.05

X_masked = X_norm.loc[:,var_mask]

print(X_raw.shape)
print(X_masked.shape)
```

```
(76020, 369)
(76020, 333)
```

4. Calculate pairwise predictor feature correlations. Identify and drop the predictor features with 95% and more correlation with another feature. Make sure to keep one of the redundant features. Assign the reduced dataset back into X. Display the shape of the resulting X.

```
corr_matrix = X_masked.corr().abs()

mask = np.triu(np.ones_like(corr_matrix, dtype = bool))
tri_df = corr_matrix.mask(mask)

to_drop = [c for c in tri_df.columns if any(tri_df[c] > 0.95)]

X_reduced = X_masked.drop(to_drop, axis=1)

print(X_masked.shape)
print(X_reduced.shape)
```

```
(76020, 333)
(76020, 204)
```

Selection Using Model Performance

5. Use StandardScaler( ) to standardize the predictor dataset (X). Next, use logistic regression with L1 regularization to identify most relevant features in predicting the target feature.

Create a mask to represent the selected features (with coefficient of more than zero). Display

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler


stsc = StandardScaler()
X_scaled = pd.DataFrame(stsc.fit_transform(X_reduced), columns = X_reduced.columns)


lr = LogisticRegression(penalty="l1", solver="saga", n_jobs=-1, max_iter=100000)
lr.fit(X_scaled, y)
## Runtime: 15m
```

```
LogisticRegression(max_iter=100000, n_jobs=-1, penalty='l1', solver='saga')
```

```
lr_mask = lr.coef_[0] > 0
print(lr_mask)
print(f"Total features: {len(lr_mask)} \nSelected features: {sum(lr_mask)}")
```

```
[False  True  True  True False False  True False  True False False False
  True False False False False False  True False  True False False False
  True  True False False False  True  True  True  True  True  True False
 False False False False  True False False False False False False  True
 False False  True  True False False False False False  True False False
 False False False  True False  True  True False  True  True False False
 False False False False False False False False False False  True  True
  True  True  True False False  True False False  True False False False
 False False False False False False False False False False  True  True
  True False False False False False False  True False  True False False
 False False False False False False False False False False False  True
  True  True  True  True False False False  True False False False False
 False  True False False False False  True False  True False  True False
 False  True False  True False False False False False False False False
 False False False False  True  True  True False False False  True False
 False False  True  True False False  True False False False False False
 False False False False False False False False False False False False]
Total features: 204
Selected features: 57
```

6. Continue with your standardized predictor dataset (X). Use SVM with L1 regularization to identify most relevant features in predicting the target feature.

   Create a mask to represent the selected features (with coefficient of more than zero). Display the number of features selected by this method.

```
from sklearn.svm import LinearSVC


lsvc = LinearSVC(penalty="l1", dual = False, max_iter=100000)
```

```
lsvc.fit(X_scaled, y)
## Runtime: 14m
```

```
LinearSVC(dual=False, max_iter=100000, penalty='l1')
```

```
lsvc_mask = lsvc.coef_[0] > 0
print(lsvc_mask)
print(f"Total features: {len(lsvc_mask)} \nSelected features: {sum(lsvc_mask)}")
```

```
[False  True  True  True False False  True False  True False  True False
  True False False  True False  True  True  True  True False False False
  True  True False False  True False False  True  True  True  True False
 False False False  True  True False False False False False False  True
 False False  True  True False False False False False  True False False
 False False False False False  True  True False  True  True False False
 False False False False False False False False False False False  True
  True  True  True False False  True False False  True False False False
 False False False False False False False False False False  True False
  True False False False False False False  True False  True False False
 False False False False False False False False False False False  True
  True False  True  True  True False  True  True  True False False False
 False  True False False  True  True False False  True False  True False
 False  True False  True  True False False False False False False False
 False False False False  True  True False False False False  True False
 False False  True  True False False  True False False False False False
 False False False False False False False False False False False False]
Total features: 204
Selected features: 61
```

7. Use your non-standardized predictor dataset (X). Use Random Forest to identify most relevant features in predicting the target feature. It means features with importance of let's say features with over 0.1% of feature importance.

   Create a mask to represent the selected features. Display the number of features selected by this method.

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier(min_samples_leaf=.0001)
rf.fit(X_scaled, y)
```

```
RandomForestClassifier(min_samples_leaf=0.0001)
```

```
rf_mask = rf.feature_importances_ > 0.001
print(rf_mask)
print(f"Total features: {len(rf_mask)} \nSelected features: {sum(rf_mask)}")
```

```
[ True  True  True False False False  True False  True  True  True  True
  True False  True False False False False False False False  True False
```

```
 False False False False  True  True  True  True  True  True  True  True
  True False  True False False False False False False False False False
 False False False  True False  True  True False  True  True False False
 False False False  True  True  True False  True False  True  True False
 False  True  True False  True False False  True False  True False  True
 False False  True False False False False False False False False False
 False False False False False False False False False False False False
  True  True False False False False False False  True  True  True  True
 False False False False False False False False False False False  True
  True  True  True  True  True  True  True False  True False False False
 False  True False False False False False  True  True  True  True False
 False False False False  True  True False False False False False False
 False False False  True  True  True  True  True  True  True  True  True
 False  True  True  True False  True  True False  True False False False
 False False False False False False False False False False False  True]
Total features: 204
Selected features: 77
```

8. Use an ensemble method based on voting to combine the above three masks. You should select features that have 3 out of 3 votes. Display the number of such features.

```
ensemble_vote = (rf_mask.astype(int) + lsvc_mask.astype(int) + lr_mask.astype(int))
print(ensemble_vote)
```

```
[1 3 3 2 0 0 3 0 3 1 2 1 3 0 1 1 0 1 2 1 2 0 1 0 2 2 0 0 2 2 2 3 3 3 3 1 1
 0 1 1 2 0 0 0 0 0 0 2 0 0 2 3 0 1 1 0 1 3 0 0 0 0 0 2 1 3 2 1 2 3 1 0 0 1
 1 0 1 0 0 1 0 1 1 3 2 2 3 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 3 1 0
 0 0 0 0 2 1 3 1 1 0 0 0 0 0 0 0 0 0 0 3 3 2 3 3 2 1 2 2 2 0 0 0 0 3 0 0
 1 1 1 1 3 1 3 0 0 2 0 2 2 1 0 0 0 0 0 0 0 0 0 1 3 3 2 1 1 1 3 1 0 1 3 3 0
 1 3 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
```

```
ensemble_mask = ensemble_vote == 3
print(ensemble_mask)
print(f"Total features: {len(ensemble_mask)} \nSelected features: {sum(ensemble_mask)}")
```

```
[False  True  True False False False  True False  True False False False
  True False False False False False False False False False False False
 False False False False False False False  True  True  True  True False
 False False False False False False False False False False False False
 False False False  True False False False False False  True False False
 False False False False False  True False False False  True False False
 False False False False False False False False False False False  True
 False False  True False False False False False False False False False
 False False False False False False False False False False False False
  True False False False False False False False False  True False False
 False False False False False False False False False False False  True
  True False  True  True False False False False False False False False
 False  True False False False False False False  True False  True False
 False False False False False False False False False False False False
 False False False False  True  True False False False False  True False
 False False  True  True False False  True False False False False False
```

```
 False False False False False False False False False False False False]
Total features: 204
Selected features: 30
```

Modeling

9. Use the selecetd features in the above question to train and test a linear SVM model and predict the target feature. Display the accuracy rate and classification report for train and test sets.

```
X_final = X_scaled.loc[:, ensemble_mask]
```

```
from sklearn.model_selection import train_test_split
```

```
seed = 24681357
X_train, X_test, y_train, y_test = train_test_split(X_final, y, test_size=0.2, random_state =
```

```
lsvc_ensemble = LinearSVC(dual = False, max_iter = 1000000)
lsvc_ensemble.fit(X_train, y_train)
```

```
LinearSVC(dual=False, max_iter=1000000)
```

```
y_train_pred = lsvc_ensemble.predict(X_train)
y_test_pred = lsvc_ensemble.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
```

```
print(accuracy_score(y_train, y_train_pred))
print(accuracy_score(y_test, y_test_pred))
```

```
0.9599611944225204
0.9623125493291239
```

1s completed at 2:40 PM