

```

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression

```

```

y = pd.read_csv('actual.csv')
print(y.shape)
y.head()

```

```

↳ (72, 2)

```

	patient	cancer
0	1	ALL
1	2	ALL
2	3	ALL
3	4	ALL
4	5	ALL

```

y['cancer'].value_counts()

```

```

ALL    47
AML    25
Name: cancer, dtype: int64

```

```

y= y.replace({'ALL':0, 'AML':1})
labels = ['ALL', 'AML']

```

```

# Import training data
df_train = pd.read_csv("data_set_ALL_AML_train.csv")
print(df_train.shape)

```

```

# Import testing data
df_test = pd.read_csv('data_set_ALL_AML_independent.csv')
print(df_test.shape)

```

```

(7129, 78)
(7129, 70)

```

```

df_train.head()

```

	Gene Description	Gene Accession Number	1	call	2	call.1	3	call.2	4	call.3	5
0	AFFX-BioB- 5_at (endogenous control)	AFFX- BioB-5_at	-214	A	-139	A	-76	A	-135	A	-106
1	AFFX-BioB- M_at (endogenous control)	AFFX- BioB-M_at	-153	A	-73	A	-49	A	-114	A	-125
2	AFFX-BioB- 3_at (endogenous control)	AFFX- BioB-3_at	-58	A	-1	A	-307	A	265	A	-76
3	AFFX-BioC- 5_at (endogenous control)	AFFX- BioC-5_at	88	A	283	A	309	A	12	A	168
4	AFFX-BioC- 3_at (endogenous control)	AFFX- BioC-3_at	-295	A	-264	A	-376	A	-419	A	-230

```
df_test.head()
```

```

    Gene      Gene
    Description Accession 39 call 40 call.1 42 call.2 47 call.3 48

# Remove "call" columns from training and testing data
train_to_keep = [col for col in df_train.columns if "call" not in col]
test_to_keep = [col for col in df_test.columns if "call" not in col]

X_train_tr = df_train[train_to_keep]
X_test_tr = df_test[test_to_keep]

4      M_at      AFFX- 200      A 248      A 152      A 248      A 177
train_columns_titles = ['Gene Description', 'Gene Accession Number', '1', '2', '3', '4', '
    '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '
    '26', '27', '28', '29', '30', '31', '32', '33', '34', '35', '36', '37', '38']

X_train_tr = X_train_tr.reindex(columns=train_columns_titles)

test_columns_titles = ['Gene Description', 'Gene Accession Number', '39', '40', '41', '42',
    '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', '58', '59',
    '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72']

X_test_tr = X_test_tr.reindex(columns=test_columns_titles)

4      M_at      AFFX- 224      A -226      A -211      A -289      A -170
X_train = X_train_tr.T
X_test = X_test_tr.T

print(X_train.shape)
X_train.head()
```

(40, 7129)

	0	1	2	3	4	
Gene	AFFX-BioB-	AFFX-BioB-	AFFX-BioB-	AFFX-BioC-	AFFX-BioC-	AFFX-
Description	5_at	M_at	3_at	5_at	3_at	BioDn-5_
	(endogenous	(endogenous	(endogenous	(endogenous	(endogenous	(endogenous
	control)	control)	control)	control)	control)	control)
Gene	AFFX-BioB-	AFFX-BioB-	AFFX-BioB-	AFFX-BioC-	AFFX-BioC-	AFFX-
Accession	5_at	M_at	3_at	5_at	3_at	BioDn-5_
Number						
1	-214	-153	-58	88	-295	-58
2	-139	-73	-1	283	-264	-40
3	-76	-49	-307	309	-376	-68

5 rows × 7129 columns

```

# Clean up the column names for training and testing data
X_train.columns = X_train.iloc[1]
X_train = X_train.drop(["Gene Description", "Gene Accession Number"]).apply(pd.to_numeric)
```

```
# Clean up the column names for Testing data
X_test.columns = X_test.iloc[1]
X_test = X_test.drop(["Gene Description", "Gene Accession Number"]).apply(pd.to_numeric)

print(X_train.shape)
print(X_test.shape)
X_train.head()
```

```
(38, 7129)
```

```
(34, 7129)
```

Gene Accession Number	AFFX- BioB- 5_at	AFFX- BioB- M_at	AFFX- BioB- 3_at	AFFX- BioC- 5_at	AFFX- BioC- 3_at	AFFX- BioDn- 5_at	AFFX- BioDn- 3_at	AFFX- CreX- 5_at	AFFX- CreX- 3_at	AFFX- BioB- 5_st	AFFX- BioB- 3_st
1	-214	-153	-58	88	-295	-558	199	-176	252	206	
2	-139	-73	-1	283	-264	-400	-330	-168	101	74	
3	-76	-49	-307	309	-376	-650	33	-367	206	-215	
4	-135	-114	265	12	-419	-585	158	-253	49	31	
5	-106	-125	-76	168	-230	-284	4	-122	70	252	

```
5 rows × 7129 columns
```

```
# Subset the first 38 patient's cancer types
X_train = X_train.reset_index(drop=True)
y_train = y[y.patient <= 38].reset_index(drop=True)
```

```
# Subset the rest for testing
X_test = X_test.reset_index(drop=True)
y_test = y[y.patient > 38].reset_index(drop=True)
```

```
log_grid = {'C': [1e-03, 1e-2, 1e-1, 1, 10],
            'penalty': ['l1', 'l2']}
```

```
log_estimator = LogisticRegression(solver='liblinear')
```

```
log_model = GridSearchCV(estimator=log_estimator,
                          param_grid=log_grid,
                          cv=3,
                          scoring='accuracy')
```

```
log_model.fit(X_train, y_train.iloc[:,1])
```

```
print("Best Parameters:\n", log_model.best_params_)
```

```
# Select best log model
best_log = log_model.best_estimator_
```

```
# Make predictions using the optimised parameters
log_pred = best_log.predict(X_test)
```

```
print('Logistic Regression accuracy:', round(accuracy_score(y_test.iloc[:,1], log_pred), 3)  
cm_log = confusion_matrix(y_test.iloc[:,1], log_pred)
```

```
ax = plt.subplot()  
sns.heatmap(cm_log, annot=True, ax = ax, fmt='g', cmap='Greens')
```

```
# labels, title and ticks  
ax.set_xlabel('Predicted labels')  
ax.set_ylabel('True labels')  
ax.set_title('Logistic Regression Confusion Matrix')  
ax.xaxis.set_ticklabels(labels)  
ax.yaxis.set_ticklabels(labels, rotation=360);
```

Best Parameters:

{'C': 0.1, 'penalty': 'l1'}

Logistic Regression accuracy: 1.0



