Import

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import f1_score, accuracy_score, roc_auc_score
```

Data

```
X, Y = load_breast_cancer(return_X_y=True)
```

```
print(X.shape)
print(type(X))
print(Y.shape)
```

```
    (569, 30)
    <class 'numpy.ndarray'>
    (569,)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, stratify=Y, test_size=0.3)
print(X_train.shape)
print(X_test.shape)
```

```
    (398, 30)
    (171, 30)
```

Simple Model

```
model = LogisticRegression(penalty='l2', C=1, n_jobs=-1)
model.fit(X_train, Y_train)
```

```
    LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                       intercept_scaling=1, l1_ratio=None, max_iter=100,
                       multi_class='auto', n_jobs=-1, penalty='l2',
                       random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                       warm_start=False)
```

```
model.class_weight
```

```
model.classes_
```

```
    array([0, 1])
```

```
model.coef_
```

```
    array([[ 0.50446288,  0.46536999,  0.47822535, -0.025185  , -0.01637015,
            -0.08412742, -0.11147375, -0.04604839, -0.02704478, -0.00586352,
             0.01599791,  0.16761935,  0.0846162 , -0.13037435, -0.00105696,
            -0.01953819, -0.02387281, -0.00643695, -0.00795846, -0.00186939,
             0.47588148, -0.49645647, -0.26039859, -0.0089375 , -0.03264828,
            -0.28722697, -0.33186061, -0.10306915, -0.09759677, -0.02841743]])
```

```
model.predict_proba(X_test[0,:].reshape(1,-1))
```

```
    array([[0.00306268, 0.99693732]])
```

```
model.predict(X_test[0].reshape(1,-1))
```

```
    array([1])
```

```
model.predict_log_proba(X_test[0].reshape(1,-1))
```

```
    array([[-5.78846453e+00, -3.06738081e-03]])
```

Hyper parameter tuning

```
C = np.array([0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.5, 0.1, 1, 2, 4, 8, 16,
model = LogisticRegression()
```

```
clf = RandomizedSearchCV(model, {'C':lambdas}, n_iter=20 if len(lambdas)>20 else len(lambda
clf.fit(X_train, Y_train)
```

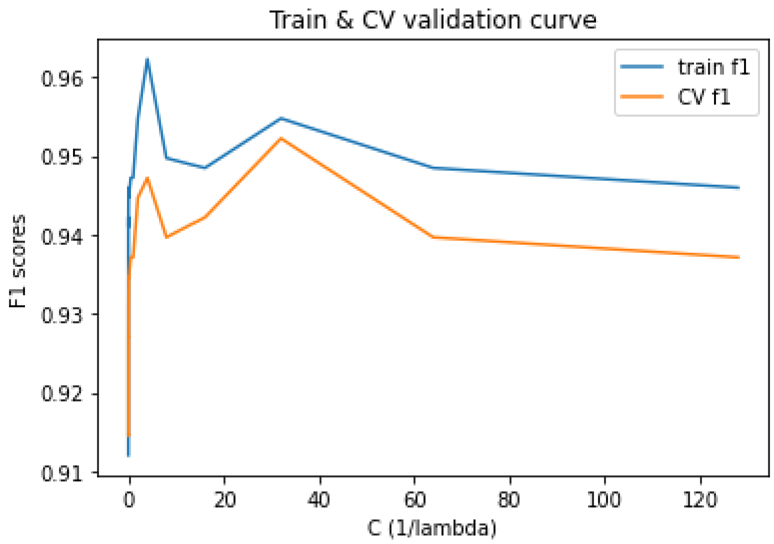
```
    /usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converge
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
      extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
    RandomizedSearchCV(cv=3, error_score='f1',
                       estimator=LogisticRegression(C=1.0, class_weight=None,
                                                    dual=False, fit_intercept=True,
                                                    intercept_scaling=1,
                                                    l1_ratio=None, max_iter=100,
                                                    multi_class='auto', n_jobs=None,
                                                    penalty='l2', random_state=None,
                                                    solver='lbfgs', tol=0.0001,
                                                    verbose=0, warm_start=False),
                       iid='deprecated', n_iter=17, n_jobs=-1,
                       param_distributions={'C': array([1.00e-05, 5.00e-04, 1.00e-04, 5.00
           1.00e-02, 5.00e-01, 1.00e-01, 1.00e+00, 2.00e+00, 4.00e+00,
           8.00e+00, 1.60e+01, 3.20e+01, 6.40e+01, 1.28e+02])},
                       pre_dispatch='2*n_jobs', random_state=None, refit=True,
                       return_train_score=True, scoring=None, verbose=0)
```

```
results = clf.cv_results_
results = pd.DataFrame.from_dict(results)
results.sort_values('param_C', inplace=True)
results
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	p
0	0.039121	0.004723	0.000592	0.000010	1e-05	{
2	0.039807	0.008152	0.000631	0.000089	0.0001	(
1	0.049932	0.005235	0.000588	0.000019	0.0005	(
4	0.047818	0.002255	0.000709	0.000155	0.001	
3	0.046624	0.002617	0.000611	0.000030	0.005	
6	0.046301	0.001798	0.000562	0.000008	0.01	
5	0.046118	0.001151	0.000591	0.000032	0.05	
8	0.055130	0.003343	0.000608	0.000031	0.1	
7	0.049666	0.003256	0.000662	0.000108	0.5	
9	0.051179	0.000403	0.000598	0.000040	1	
10	0.045281	0.001788	0.000586	0.000006	2	
11	0.056667	0.013879	0.000590	0.000035	4	
12	0.057454	0.006928	0.001780	0.001662	8	
13	0.052055	0.005463	0.000588	0.000022	16	
14	0.046962	0.000577	0.000618	0.000045	32	
15	0.048197	0.004962	0.000591	0.000030	64	
16	0.043371	0.004832	0.000541	0.000107	128	

```
plt.plot(results['param_C'], results['mean_train_score'], label='train f1')
plt.plot(results['param_C'], results['mean_test_score'], label='CV f1')
plt.xlabel("C (1/lambda)")
plt.ylabel("F1 scores")
plt.title("Train & CV validation curve")
plt.legend()
plt.show()
```



```
low_score = results[((results['mean_train_score']-results['mean_test_score']) > 0)]
low_score = low_score[]
```

```
arg = (low_score['mean_train_score']-low_score['mean_test_score']).argmin()
best_C = low_score.iloc[arg]['param_C']
```

```
model = LogisticRegression(C=best_C, max_iter=1000)
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
Y_pred_proba = model.predict_proba(X_test)
print("F1 score :", f1_score(Y_test, Y_pred))
print("AUC score :", roc_auc_score(Y_test, Y_pred_proba[:,1]))
print("Accuracy score :", accuracy_score(Y_test, Y_pred))
```

```
F1 score : 0.9626168224299065
AUC score : 0.9924065420560748
Accuracy score : 0.9532163742690059
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Just analysis on C

```
%%time
model = LogisticRegression(C=1, penalty="l1", solver='liblinear')
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
print(np.count_nonzero(model.coef_))
```

```
10
CPU times: user 177 ms, sys: 0 ns, total: 177 ms
Wall time: 186 ms
```

```
%%time
```

```
model = LogisticRegression(C=0.1, penalty="l1", solver='liblinear')
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
print(np.count_nonzero(model.coef_))
```

```
6
CPU times: user 102 ms, sys: 0 ns, total: 102 ms
Wall time: 103 ms
```

```
%%time
model = LogisticRegression(C=0.001, penalty="l1", solver='liblinear')
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
print(np.count_nonzero(model.coef_))
```

```
3
CPU times: user 12.5 ms, sys: 0 ns, total: 12.5 ms
Wall time: 12.5 ms
```

More and more underfit as λ increases (i.e.) as C decreases. It is evident via no of non zero elements in the weight vector

✓ 0s completed at 3:35 PM ● ×