from sklearn.linear\_model import LogisticRegression

seed(0)

# skf = StratifiedKFold(n\_splits=5)

params0 = {'tol' : [1e-6,1e-5,1e-4,1e-3,1e-2],

'C': [0.5,1.0,1.5,2.0,2.5]}

lg = LogisticRegression(random\_state=0, solver='lbfgs',multi\_class='multinomial')

lg = GridSearchCV(lg, cv=5, param\_grid=params0, scoring = 'roc\_auc',refit = True,

n\_jobs=-1, verbose = 5, return\_train\_score=True)

lg.fit(X\_train, y\_train)

lg.cv\_results\_

————————————————————————————————————————

lg.best\_estimator\_

————————————————————————————————————————

lg\_pred = lg.best\_estimator\_.predict(X\_test)

lg\_prob = lg.best\_estimator\_.predict\_proba(X\_test)

lg\_prob

————————————————————————————————————————

lg\_matrix = metrics.confusion\_matrix(y\_test, lg\_pred)

lg\_matrix

————————————————————————————————————————

lg\_test = lg.best\_estimator\_.score(X\_test, y\_test)

lg\_matrix = metrics.confusion\_matrix(y\_test, lg\_pred)

lg\_cm = pd.DataFrame(lg\_matrix, range(2), range(2))

# plt.figure(figsize=(5, 8))

fig, ax = plt.subplots(figsize=(6,4))

akws = {"ha": 'center',"va": 'center','c':'black','fontsize':'20'}

ax = sns.heatmap (lg\_cm, fmt='d',

cmap='Oranges\_r', annot=True, square = True,ax=ax,linewidths=0.5,annot\_kws=akws)

bottom, top = ax.get\_ylim()

ax.set\_ylim(bottom + 0.5, top - 0.5)

plt.xlabel('Predicted Class')

plt.ylabel('Actual Class')

lg\_title = 'Logistic Regression - Confussion Matrix on Test Data \nMean Accuracy Score: {0:2f}'.format(lg\_test)

plt.title(lg\_title, size = 14)

# plt.figure(figsize=(16, 26))

plt.show;

————————————————————————————————————————

print("", classification\_report(y\_test, lg\_pred, target\_names=target\_names))

————————————————————————————————————————

acc\_lg = accuracy\_score(y\_test, lg\_pred)

print("Logistic Regression accuracy:", acc\_lg)

————————————————————————————————————————

lg\_probs = lg.best\_estimator\_.predict\_proba(X\_test)[:,1]

print(roc\_auc\_score(y\_test, lg\_probs))

————————————————————————————————————————

error\_lg = 1-acc\_lg

error\_lg

————————————————————————————————————————

For training set:

————————————————————————————————————————

lg\_pred\_tr = lg.best\_estimator\_.predict(X\_train)

lg\_prob\_tr = lg.best\_estimator\_.predict\_proba(X\_train)

lg\_prob\_tr

————————————————————————————————————————

lg\_matrix\_tr = metrics.confusion\_matrix(y\_train, lg\_pred\_tr)

lg\_matrix\_tr

————————————————————————————————————————

lg\_train = lg.best\_estimator\_.score(X\_train, y\_train)

lg\_matrix = metrics.confusion\_matrix(y\_train, lg\_pred\_tr)

lg\_cm\_tr = pd.DataFrame(lg\_matrix, range(2), range(2))

fig, ax = plt.subplots(figsize=(6,4))

akws = {"ha": 'center',"va": 'center','c':'red','fontsize':'20'}

ax = sns.heatmap (lg\_cm\_tr, fmt='d',

cmap='Blues\_r', annot=True, square = True,ax=ax,linewidths=0.5,annot\_kws=akws)

bottom, top = ax.get\_ylim()

ax.set\_ylim(bottom + 0.5, top - 0.5)

plt.xlabel('Predicted Class')

plt.ylabel('Actual Class')

lg\_title = 'Logistic Regression - Confussion Matrix on Train Data \nMean Accuracy Score: {0:2f}'.format(lg\_train)

plt.title(lg\_title, size = 14)

# plt.figure(figsize=(16, 26))

plt.show;

————————————————————————————————————————————

print("", classification\_report(y\_train, lg\_pred\_tr, target\_names=target\_names))

————————————————————————————————————————

y\_pred = lg\_probs

y\_true = y\_test\_v

print("Original ROC area: {:0.4f}".format(roc\_auc\_score(y\_true, y\_pred)))

n\_bootstraps = 1000

rng\_seed = 42 # control reproducibility

bootstrapped\_scores = []

rng = np.random.RandomState(rng\_seed)

for i in range(n\_bootstraps):

# bootstrap by sampling with replacement on the prediction indices

indices = rng.randint(0, len(y\_pred), len(y\_pred))

if len(np.unique(y\_true[indices])) < 2:

# We need at least one positive and one negative sample for ROC AUC

# to be defined: reject the sample

continue

score = roc\_auc\_score(y\_true[indices], y\_pred[indices])

bootstrapped\_scores.append(score)

#print("Bootstrap #{} ROC area: {:0.3f}".format(i + 1, score))

import matplotlib.pyplot as plt

plt.hist(bootstrapped\_scores, bins=50)

plt.title('Histogram of the bootstrapped ROC AUC scores')

plt.show()

sorted\_scores = np.array(bootstrapped\_scores)

sorted\_scores.sort()

# Computing the lower and upper bound of the 90% confidence interval

# You can change the bounds percentiles to 0.025 and 0.975 to get

# a 95% confidence interval instead.

confidence\_lower = sorted\_scores[int(0.05 \* len(sorted\_scores))]

confidence\_upper = sorted\_scores[int(0.95 \* len(sorted\_scores))]

print("Confidence interval for the score: [{:0.4f} - {:0.4}]".format(

confidence\_lower, confidence\_upper))

————————————————————————————————————————

alpha = .95

y\_pred = lg\_probs

y\_true = y\_test\_v2

auc, auc\_cov = delong\_roc\_variance(

y\_true,

y\_pred)

auc\_std = np.sqrt(auc\_cov)

lower\_upper\_q = np.abs(np.array([0, 1]) - (1 - alpha) / 2)

ci = stats.norm.ppf(

lower\_upper\_q,

loc=auc,

scale=auc\_std)

ci[ci > 1] = 1

print('AUC:', auc)

print('AUC COV:', auc\_cov)

print('95% AUC CI:', ci)

————————————————————————————————————————

from matplotlib import pyplot

print(dt.best\_estimator\_.feature\_importances\_)

pyplot.bar(range(len(dt.best\_estimator\_.feature\_importances\_)), dt.best\_estimator\_.feature\_importances\_)

pyplot.show()

————————————————————————————————————————

feat\_importances\_dt = pd.Series(dt.best\_estimator\_.feature\_importances\_, index=X\_train.columns)

feat\_importances\_dt.nlargest(5).plot(kind='barh')

pyplot.show()