```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon May 16 15:00:39 2022

@author: hillarywolff
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
%matplotlib inline
import seaborn as sns
sns.set()
sns.set_style("darkgrid")
import warnings
warnings.filterwarnings("ignore")
import math
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import cross_val_score, GridSearchCV,
KFold, train_test_split
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

def vector_values(grid_search, trials):
    mean_vec = np.zeros(trials)
    std_vec = np.zeros(trials)
    i = 0
    final = grid_search.cv_results_

    #Using Grid Search's 'cv_results' attribute to get mean and std
for each paramter
    for mean_score, std_score in zip(final["mean_test_score"],
final["std_test_score"]):
        mean_vec[i] = -mean_score
        # negative sign used with mean.score() to get positive mean
squared error
        std_vec[i] = std_score
        i = i+1

    return mean_vec, std_vec

# filter variables

PATH = r"/Users/hillarywolff/Documents/GitHub/machine_learning/MP3/"
df = pd.read_csv(PATH + 'Covid002.csv', encoding='latin-1')

col_use = pd.read_excel(PATH+'Variable Description.xlsx')
```

```
var_use = col_use[(col_use['Source'] == 'Opportunity Insights') |
(col_use['Source']=='PM_COVID')]

var_use = list(var_use['Variable'])
var_use.extend(['county', 'state', 'deathspc'])
df = df[var_use]

# 2. compute descriptive statistics for the subset of variables

summary = df.describe()

# 3 drop na
df = df.dropna()

# 4. create separate dummy for each state and DC
df = pd.get_dummies(df, columns =['state'])

# 5. split the sample into training and test set (80% and 20%)
X = df[df.columns.difference(['deathspc', 'county'])]
y = df[['deathspc']]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=25)

# using training set, estimate relationship between COVID_19 deaths
per capita
# and the opportunity insights and PM COVID predictors as well as
state level
# fixed effects

reg = LinearRegression().fit(X_train, y_train)
y_pred = reg.predict(X_test)
mean_squared_error(y_test, y_pred)
# 1927.2
y_pred = reg.predict(X_train)
mean_squared_error(y_train, y_pred)
#1561.85


# a. based on those estimates, calculate and report the MSE in both
training
# and validation sets
y_pred = reg.predict(X_test)
mean_squared_error(y_test, y_pred)
# 1982.75

# b. why might you be concerned with overfitting in this context? is
there any
# evidence of overfitting?
# since p is less than n, we are not concerned for overfitting
```

```
outright, but
# it is likely that there are some extraneous variables being used in
our
# model which could overfit our data.




# 7. use the training set to estimate ridge regression and lasso
analogs to the
# OLS model in the previous question. For each, you should report a
plot of the
# cross validation estimates of the test error as a function of the
hyperparameter
# that indicates the tuned value of lambda.
ridge = Ridge(normalize = True)
alpha_param = (10**np.linspace(start=-2, stop=2, num=100))

param_grid = [{'alpha': alpha_param }]

grid_search_ridge = GridSearchCV(ridge, param_grid, cv = 10, scoring =
'neg_mean_squared_error')
grid_search_ridge.fit(X, y)

mean_vec, std_vec = vector_values(grid_search_ridge, 100)

plt.figure(figsize=(12,10))
plt.title('Ridge Regression', fontsize= 20)
plt.plot(np.log(alpha_param), mean_vec)
plt.errorbar(np.log(alpha_param), mean_vec, yerr = std_vec)
plt.ylabel("MSE", fontsize= 20)
plt.xlabel("log(Alpha)", fontsize= 20)
plt.show()

print(min(mean_vec))
# 2303.30

alpha_ridge = alpha_param[np.where(mean_vec == min(mean_vec))][0]
# 0.135

clf = Ridge(normalize=True, alpha=alpha_ridge)
clf.fit(X, y)



# lasso
lasso = Lasso(normalize = True)
alpha_param = (10**np.linspace(start=-2, stop=2, num=100))

param_grid = [{'alpha': alpha_param }]
```

```
grid_search_lasso = GridSearchCV(lasso, param_grid, cv = 10, scoring =
'neg_mean_squared_error')
grid_search_lasso.fit(X, y)


mean_vec, std_vec = vector_values(grid_search_lasso, 100)

plt.figure(figsize=(12,10))
plt.title('Lasso Regression', fontsize= 20)
plt.plot(np.log(alpha_param), mean_vec)
plt.errorbar(np.log(alpha_param), mean_vec, yerr = std_vec)
plt.ylabel("MSE", fontsize= 20)
plt.xlabel("log(Alpha)", fontsize= 20)
plt.show()


print(min(mean_vec))
# 2277.01

alpha_lasso = alpha_param[np.where(mean_vec == min(mean_vec))][0]
# 0.01

lass = Lasso(normalize=True, alpha=alpha_lasso)
lass.fit(X, y)



# MSE Ridge test
y_pred = clf.predict(X_test)
mean_squared_error(y_test, y_pred)
# 1701

# MSE ridge train
y_pred = clf.predict(X_train)
mean_squared_error(y_train, y_pred)
# 1618.85

# MSE Lasso test
y_pred = lass.predict(X_test)
mean_squared_error(y_test, y_pred)
# 1663.49

# MSE lasso train
y_pred = lass.predict(X_train)
mean_squared_error(y_train, y_pred)
# 1653.23

# 8. did ridge regression and or the lasso improve your prediction
over OLS?
# the ridge and lasso regressions improved our prediction over OLS and
# our lasso regression is the method I would recommend since it has
the lowest MSE
# as well as the lowest difference between our validation and test
predictions.
# ridge would be second best and OLS would be the worst. This is
because we are
# looking for the lowest MSE since it balances bias and varaince well
```