Task 3

In [2]:
```
import warnings

warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
from sklearn import linear_model


def regression(data_train, data_test):


    X_train = data_train.drop(columns=['price'])
    y_train = data_train['price']

    reg = linear_model.LassoCV(alphas=np.logspace(-4, -1, 4), normalize=True).fit(X_train, y_train)

    X_test = data_test.drop(columns=['price'])
    y_test = data_test['price']

    pred_lasoo = np.round(reg.predict(X_test), 2)
    alpha_lasoo = reg.alpha_
    coef_lasoo = pd.DataFrame()
    coef_lasoo['variable'] = X_train.columns
    coef_lasoo['coef'] = reg.coef_
    coef_lasoo = coef_lasoo[coef_lasoo['coef'] != 0]

    reg = linear_model.RidgeCV(alphas=np.logspace(-4, -1, 4), normalize=True).fit(X_train, y_train)
    pred_ridge = np.round(reg.predict(X_test),2)
    alpha_ridge = reg.alpha_
    coef_ridge = pd.DataFrame()
    coef_ridge['variable'] = X_train.columns
    coef_ridge['coef'] = reg.coef_
    coef_ridge = coef_ridge[abs(coef_ridge['coef']) > 0.001]

    reg = linear_model.ElasticNetCV(l1_ratio = np.arange(0.6, 1, 0.1), alphas=np.logspace(-4, -1, 4), normalize=True).fit(X_train, y_train)
    pred_elastic = np.round(reg.predict(X_test),2)
    alpha_elastic = reg.alpha_
    coef_elastic = pd.DataFrame()
    coef_elastic['variable'] = X_train.columns
    coef_elastic['coef'] = reg.coef_
    coef_elastic = coef_elastic[abs(coef_elastic['coef']) > 0.001]
    l1_ratio_elastic = reg.l1_ratio_


    return {'ridge': {'alpha': alpha_ridge, 'pred': pred_ridge, 'coefficients': coef_ridge},
            'lasso': {'alpha': alpha_lasoo, 'pred': pred_lasoo, 'coefficients': coef_lasoo},
            'elastic_net': {'alpha': alpha_elastic, 'l1_ratio': l1_ratio_elastic, 'pred': pred_elastic, 'coefficients': coef_elastic}
            }

data_train = pd.read_csv("data_train.csv")
data_test = pd.read_csv("data_test.csv")
regression(data_train, data_test)
```

Out[2]:
```
{'ridge': {'alpha': 0.001,
  'pred': array([  1.27, 536.35, 384.7 , 131.77, 166.29, 104.64, 350.55, 262.11,
         338.08, 169.29]),
  'coefficients':    variable      coef
  0       S_0  0.001791
  1       S_1 -0.009615
  2       S_2  0.008826
  3       S_3  1.427210
  4       S_4 -0.019536
  ..      ...       ...
  91     S_91  0.005885
  92     S_92 -0.006339
  93     S_93 -0.155255
  94     S_94  0.083996
  95     S_95  0.003478

  [75 rows x 2 columns]},
 'lasso': {'alpha': 0.01,
  'pred': array([  6.22, 531.25, 377.35, 134.61, 154.21, 102.41, 362.97, 263.1 ,
         334.38, 167.29]),
  'coefficients':    variable       coef
  1       S_1  -0.003165
  2       S_2   0.011310
  3       S_3   1.363747
  4       S_4  -0.011285
  5       S_5  13.108086
  6       S_6   0.000097
  10     S_10   4.660084
  14     S_14   0.007331
  16     S_16   0.019221
  17     S_17  -0.000392
  18     S_18   0.003064
  19     S_19  -0.002484
  28     S_28  -5.570983
  30     S_30  -0.001902
  33     S_33   0.000391
  34     S_34   2.228081
  41     S_41  -0.003377
  42     S_42   0.000185
  47     S_47   6.825045
  49     S_49  -0.003673
  52     S_52   0.000138
  62     S_62   0.003197
  64     S_64  -0.275039
  65     S_65  -3.863479
  70     S_70   0.000805
  71     S_71   0.003954
  73     S_73   0.002747
  80     S_80   0.001015
  83     S_83   0.886578
  86     S_86  -0.183654
  90     S_90  -0.027394
  91     S_91   0.004210
  92     S_92  -0.000994
  95     S_95   0.000515},
 'elastic_net': {'alpha': 0.0001,
  'l1_ratio': 0.7999999999999999,
  'pred': array([  1.15, 524.72, 383.27, 137.45, 163.96, 105.09, 358.82, 260.85,
         338.85, 167.65]),
  'coefficients':    variable       coef
  1       S_1  -0.007760
  2       S_2   0.014933
  3       S_3   1.268811
  4       S_4   0.001012
  5       S_5  13.216339
  ..      ...        ...
  90     S_90  -0.023281
  91     S_91   0.003718
  92     S_92  -0.004415
  93     S_93  -0.079128
  95     S_95   0.002217

  [68 rows x 2 columns]}}
```

Task 2

In [7]:
```
import pandas as pd
from scipy import stats
import numpy as np

def perform_tests(data):

    # separate numerical and categorical variables
    categorical_variable = []
    numerical_variable = []
    df = data
    for i in df.columns:
        if str(df[i].dtype) == 'int64':
            categorical_variable.append(i)
        else:
            numerical_variable.append(i)

    death_1 = df[df['death'] == 1]
    death_0 = df[df['death'] == 0]

    shapiro = []

    for column in numerical_variable:

        # test for death = 0
        shapiro_test_0 = stats.shapiro(death_0[column].values)

        # test for death = 1
        shapiro_test_1 = stats.shapiro(death_1[column].values)

        shapiro.append((column, (round(shapiro_test_0[1],4), round(shapiro_test_1[1], 4))))


    # chi-sqaured
    chi = []
    for column in categorical_variable:
        if column == "death":
            continue
        else:
            crosstab = pd.crosstab(df[column], df['death'])
            res = stats.chi2_contingency(crosstab)
            chi.append((column, round(res[1], 4)))


    Whitney_values = []
    ttest_values = []
    for column in numerical_variable:
        shapiro_values = None
        for c in shapiro:
            if c[0] == column:
                shapiro_values = c
                break

        if c[1][0] > 0.05 and c[1][1] > 0.05:
            # perform unpaired t-test
            res = stats.ttest_ind(death_1[column].values, death_0[column].values, equal_var=False)
            ttest_values.append((column, round(res[1], 4)))
        else:
            # perform Mann-Whitney test
            U1, p = stats.mannwhitneyu(death_1[column].values, death_0[column].values)
            Whitney_values.append((column, round(p, 4)))

    return {
        'mann_whitney': Whitney_values,
        'ttest': ttest_values,
        'chi_square': chi,
        'shapiro_wilk': shapiro
    }

data = pd.read_csv("medical_data.csv")
perform_tests(data)
```

Out[7]:
```
{'mann_whitney': [('HDL', 0.0),
  ('LVEF', 0.0),
  ('Na+', 0.2143),
  ('K+', 0.4284),
  ('MPV', 0.4331)],
 'ttest': [('SBP', 0.0), ('DBP', 0.0), ('PLT', 0.4739), ('LDL', 0.8392)],
 'chi_square': [('amiodarone', 0.0),
  ('loop_diuretics', 0.0),
  ('ivabradine', 0.0144),
  ('ARB', 0.2338),
  ('digoxin', 0.5185),
  ('MRA', 0.2884),
  ('heart_failure', 0.0),
  ('AOS', 0.974)],
 'shapiro_wilk': [('SBP', (0.2581, 0.5881)),
  ('DBP', (0.5272, 0.3715)),
  ('PLT', (0.2361, 0.6935)),
  ('LDL', (0.9367, 0.95)),
  ('HDL', (0.4388, 0.0006)),
  ('LVEF', (0.0, 0.0)),
  ('Na+', (0.0, 0.0071)),
  ('K+', (0.0, 0.1771)),
  ('MPV', (0.0, 0.0003))]}
```

Task 1

In [9]:
```
data_path = 'jobs.csv'
from pyspark.sql.functions import count

class SparkTask:
    def __init__(self, spark_session):
        self.job_counts_dict = None
        self.sc = spark_session.sparkContext
        self.spark = spark_session

    def group_sort(self, input_path):
        data = self.spark.read.csv(input_path).toDF('_c0', '_c1')
        data = data.groupBy('_c1').agg(count('_c0'))
        df = data.toPandas()
        df = df.sort_values(by=['count(_c0)', '_c1'], ascending=[True, True]).to_dict('list')
        result = {}
        for i in range(len(df['_c1'])):
            result.update({df['_c1'][i] : df['count(_c0)'][i]})
        return result

import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder \
                    .master('local[1]') \
                    .appName('SparkByExamples.com') \
                    .getOrCreate()
spark = SparkTask(spark)

spark.group_sort(data_path)
```

Out[9]:
```
{'job': 1,
 'Pilot': 15,
 'Teacher': 24,
 'Barista': 28,
 'Designer': 29,
 'Banker': 31,
 'Office manager': 31,
 'Software engineer': 33,
 'Dancer': 34,
 'Nurse': 34,
 'Film editor': 40}
```

In [ ]: