Final Project

March 14, 2021

Importing the data

[162]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("Women Entrepreneurship.csv", sep = ",")
df = df.drop(['No'], axis = 1)
df.head()
```

[162]:

	Country	Level of development	European Union Membership	Currency	Women Entrepreneurship Index	Entrepreneurship Index	Inflation rate	Female Labor Force Participation Rate	Gender Inequality Index	Ease of Doing Business Rank	Corporate Tax Rates %	Schooling (years)
0	Austria	Developed	Member	Euro	54.9	64.9	0.90	67.1	0.069	27	25.0	15.9
1	Belgium	Developed	Member	Euro	63.6	65.5	0.60	58.0	0.043	46	25.0	16.6
2	Estonia	Developed	Member	Euro	55.4	60.2	-0.88	68.5	0.086	18	20.0	16.5
3	Finland	Developed	Member	Euro	66.4	65.7	-0.20	67.7	0.047	20	20.0	17.0
4	France	Developed	Member	Euro	68.8	67.3	0.00	60.6	0.049	32	32.0	16.3

[163]:
```
df.describe(include ='all')
```

[163]:

	Country	Level of development	European Union Membership
count	51	51	51
unique	51	2	2
top	Austria	Developed	Not Member
freq	1	27	31
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

	Currency	Women Entrepreneurship Index
count	51	51.000000
unique	2	NaN
top	National Currency	NaN
freq	36	NaN
mean	NaN	47.835294
std	NaN	14.268480
min	NaN	25.300000
25%	NaN	36.350000
50%	NaN	44.500000
75%	NaN	59.150000
max	NaN	74.800000

	Entrepreneurship Index	Inflation rate
count	51.000000	51.000000
unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	47.241176	2.587647
std	16.193149	5.380639
min	24.800000	-2.250000
25%	31.900000	-0.500000
50%	42.700000	0.600000
75%	65.400000	3.600000
max	77.600000	26.500000

	Female Labor Force Participation Rate	Gender Inequality Index
count	51.000000	51.000000
unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	58.481765	0.203216
std	13.864567	0.149461
min	13.000000	0.025000

25%	55.800000	0.069000
50%	61.000000	0.149000
75%	67.400000	0.325000
max	82.300000	0.538000

	Ease of Doing Business Rank	Corporate Tax Rates %	Schooling (years)
count	51.000000	51.000000	51.000000
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	53.137255	23.106863	15.300000
std	40.391593	5.659137	2.206717
min	2.000000	9.000000	8.000000
25%	21.500000	20.000000	13.700000
50%	40.000000	24.000000	15.400000
75%	77.000000	25.500000	16.550000
max	157.000000	34.000000	20.100000

We separate features as X and label as y

[164]:
```
X = df.drop(['Women Entrepreneurship Index'], axis = 1)
y = df['Women Entrepreneurship Index']
```

[372]:
```
X.head(50)
```

[372]:

	Country	Level of development	European Union Membership	\
0	Austria	Developed	Member	
1	Belgium	Developed	Member	
2	Estonia	Developed	Member	
3	Finland	Developed	Member	
4	France	Developed	Member	
5	Germany	Developed	Member	
6	Greece	Developed	Member	
7	Ireland	Developed	Member	
8	Italy	Developed	Member	
9	Latvia	Developed	Member	
10	Lithuania	Developed	Member	
11	Netherlands	Developed	Member	
12	Slovakia	Developed	Member	
13	Slovenia	Developed	Member	
14	Spain	Developed	Member	
15	Croatia	Developed	Member	
16	Denmark	Developed	Member	
17	Hungary	Developed	Member	
18	Poland	Developed	Member	
19	Sweden	Developed	Member	
20	Australia	Developed	Not Member	
21	Iceland	Developed	Not Member	

22	Japan	Developed	Not Member
23	Norway	Developed	Not Member
24	Singapore	Developed	Not Member
25	Switzerland	Developed	Not Member
26	Taiwan	Developed	Not Member
27	Algeria	Developing	Not Member
28	Argentina	Developing	Not Member
29	Bolivia	Developing	Not Member
30	Bosnia and Herzegovina	Developing	Not Member
31	Brazil	Developing	Not Member
32	China	Developing	Not Member
33	Costa Rica	Developing	Not Member
34	Ecuador	Developing	Not Member
35	Egypt	Developing	Not Member
36	El Salvador	Developing	Not Member
37	Ghana	Developing	Not Member
38	India	Developing	Not Member
39	Jamaica	Developing	Not Member
40	Macedonia	Developing	Not Member
41	Malaysia	Developing	Not Member
42	Mexico	Developing	Not Member
43	Panama	Developing	Not Member
44	Peru	Developing	Not Member
45	Russia	Developing	Not Member
46	Saudi Arabia	Developing	Not Member
47	Thailand	Developing	Not Member
48	Tunisia	Developing	Not Member
49	Turkey	Developing	Not Member

	Currency	Entrepreneurship Index	Inflation rate	\
0	Euro	64.9	0.90	
1	Euro	65.5	0.60	
2	Euro	60.2	-0.88	
3	Euro	65.7	-0.20	
4	Euro	67.3	0.00	
5	Euro	67.4	0.50	
6	Euro	42.0	-1.70	
7	Euro	65.3	-0.30	
8	Euro	41.3	0.00	
9	Euro	54.5	0.20	
10	Euro	54.6	-0.90	
11	Euro	66.5	0.60	
12	Euro	45.4	-0.30	
13	Euro	53.1	-0.50	
14	Euro	49.6	-0.50	
15	National Currency	40.6	-0.50	
16	National Currency	71.4	0.50	

17	National Currency	42.7	-0.10
18	National Currency	47.4	-0.90
19	National Currency	71.8	0.00
20	National Currency	77.6	1.50
21	National Currency	70.4	1.60
22	National Currency	49.5	0.80
23	National Currency	65.6	2.17
24	National Currency	68.1	-0.50
25	National Currency	68.6	-1.10
26	National Currency	69.1	-0.61
27	National Currency	30.2	4.80
28	National Currency	37.2	26.50
29	National Currency	28.0	4.10
30	National Currency	28.9	-1.00
31	National Currency	25.8	10.67
32	National Currency	36.4	1.40
33	National Currency	37.7	0.80
34	National Currency	28.2	-0.50
35	National Currency	28.1	11.00
36	National Currency	29.6	-2.25
37	National Currency	24.8	17.20
38	National Currency	25.3	5.90
39	National Currency	27.2	3.70
40	National Currency	37.1	3.70
41	National Currency	40.0	2.30
42	National Currency	30.7	2.70
43	National Currency	32.2	0.10
44	National Currency	30.9	3.50
45	National Currency	31.7	15.50
46	National Currency	49.6	1.20
47	National Currency	32.1	-0.90
48	National Currency	35.5	4.80
49	National Currency	54.6	7.70

	Female Labor Force Participation Rate	Gender Inequality Index	\
0	67.10	0.069	
1	58.00	0.043	
2	68.50	0.086	
3	67.70	0.047	
4	60.60	0.049	
5	69.90	0.084	
6	42.50	0.116	
7	59.40	0.093	
8	47.20	0.069	
9	66.40	0.176	
10	66.50	0.124	
11	69.20	0.043	

12	55.90	0.191
13	61.00	0.063
14	52.70	0.070
15	60.40	0.116
16	70.30	0.038
17	57.80	0.233
18	56.60	0.115
19	74.00	0.039
20	66.80	0.097
21	82.30	0.058
22	64.70	0.094
23	69.20	0.045
24	59.18	0.065
25	74.70	0.025
26	55.00	0.045
27	18.00	0.429
28	47.30	0.328
29	69.40	0.417
30	51.90	0.149
31	55.90	0.408
32	62.40	0.168
33	59.40	0.288
34	63.50	0.384
35	64.60	0.449
36	55.70	0.383
37	60.80	0.538
38	61.10	0.488
39	37.70	0.396
40	73.00	0.143
41	58.50	0.253
42	44.70	0.322
43	67.90	0.407
44	63.40	0.395
45	65.20	0.225
46	13.00	0.252
47	62.00	0.359
48	25.19	0.296
49	30.40	0.306

	Ease of Doing Business Rank	Corporate Tax Rates %	Schooling (years)
0	27	25.00	15.9
1	46	25.00	16.6
2	18	20.00	16.5
3	20	20.00	17.0
4	32	32.00	16.3
5	22	30.00	17.1
6	79	24.00	17.2

7	24	12.50	18.6
8	58	27.81	16.3
9	19	20.00	16.0
10	11	15.00	16.5
11	42	25.00	18.1
12	45	21.00	15.0
13	37	19.00	17.3
14	30	25.00	17.7
15	51	18.00	15.3
16	4	22.00	19.2
17	52	9.00	15.6
18	40	19.00	16.4
19	10	21.40	15.9
20	14	30.00	20.1
21	26	20.00	19.0
22	29	29.74	15.3
23	9	22.00	17.7
24	2	17.00	15.4
25	36	21.00	16.0
26	15	20.00	13.5
27	157	26.00	14.4
28	126	30.00	17.3
29	150	25.00	13.8
30	90	10.00	14.2
31	124	34.00	15.2
32	31	25.00	13.5
33	74	30.00	14.2
34	129	25.00	14.0
35	114	22.50	13.1
36	91	30.00	13.2
37	118	25.00	11.4
38	63	30.00	11.6
39	71	25.00	12.8
40	17	10.00	8.0
41	12	24.00	13.1
42	60	30.00	13.3
43	86	25.00	13.0
44	76	29.50	13.4
45	28	20.00	15.0
46	62	20.00	16.1
47	21	20.00	13.6
48	78	25.00	14.6
49	33	22.00	14.5

[166]:
```
df.describe()
```

[166]:

	Women Entrepreneurship Index	Entrepreneurship Index	Inflation rate
count	51.000000	51.000000	51.000000
mean	47.835294	47.241176	2.587647
std	14.268480	16.193149	5.380639
min	25.300000	24.800000	-2.250000
25%	36.350000	31.900000	-0.500000
50%	44.500000	42.700000	0.600000
75%	59.150000	65.400000	3.600000
max	74.800000	77.600000	26.500000

	Female Labor Force Participation Rate	Gender Inequality Index
count	51.000000	51.000000
mean	58.481765	0.203216
std	13.864567	0.149461
min	13.000000	0.025000
25%	55.800000	0.069000
50%	61.000000	0.149000
75%	67.400000	0.325000
max	82.300000	0.538000

	Ease of Doing Business Rank	Corporate Tax Rates %	Schooling (years)
count	51.000000	51.000000	51.000000
mean	53.137255	23.106863	15.300000
std	40.391593	5.659137	2.206717
min	2.000000	9.000000	8.000000
25%	21.500000	20.000000	13.700000
50%	40.000000	24.000000	15.400000
75%	77.000000	25.500000	16.550000
max	157.000000	34.000000	20.100000

Splitting the data set into Train, Test, and Validation sets

[167]:
```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(
    X_test, y_test, test_size=0.5, random_state=42)
```

[168]:
```
from pandas_profiling import ProfileReport
prof = ProfileReport(pd.concat([X_train, y_train], axis=1))

prof.to_notebook_iframe()
```

```
HBox(children=(HTML(value='Summarize dataset'), FloatProgress(value=0.0, max=26.
↪0), HTML(value='')))
```

```
HBox(children=(HTML(value='Generate report structure'), FloatProgress(value=0.0,␣
↪max=1.0), HTML(value='')))
```

```
HBox(children=(HTML(value='Render HTML'), FloatProgress(value=0.0, max=1.0),␣
↪HTML(value='')))
```

```
<IPython.core.display.HTML object>
```

Looking at the correlation of the numerical features to explore dataset and features' relationship between each other

[169]:
```
df.corr()
```

[169]:

	Women Entrepreneurship Index
Women Entrepreneurship Index	1.000000
Entrepreneurship Index	0.914580
Inflation rate	-0.455532
Female Labor Force Participation Rate	0.441372
Gender Inequality Index	-0.845388
Ease of Doing Business Rank	-0.713246
Corporate Tax Rates %	-0.199905
Schooling (years)	0.721573

	Entrepreneurship Index	Inflation rate
Women Entrepreneurship Index	0.914580	-0.455532
Entrepreneurship Index	1.000000	-0.395370
Inflation rate	-0.395370	1.000000
Female Labor Force Participation Rate	0.334170	-0.139802
Gender Inequality Index	-0.840385	0.515878
Ease of Doing Business Rank	-0.717070	0.466089
Corporate Tax Rates %	-0.166656	0.266044
Schooling (years)	0.699228	-0.197663

	Female Labor Force Participation Rate
Women Entrepreneurship Index	0.441372
Entrepreneurship Index	0.334170
Inflation rate	-0.139802
Female Labor Force Participation Rate	1.000000
Gender Inequality Index	-0.325426
Ease of Doing Business Rank	-0.360153
Corporate Tax Rates %	-0.109179
Schooling (years)	0.122918

	Gender Inequality Index

```
Women Entrepreneurship Index                    -0.845388
Entrepreneurship Index                          -0.840385
Inflation rate                                   0.515878
Female Labor Force Participation Rate           -0.325426
Gender Inequality Index                          1.000000
Ease of Doing Business Rank                      0.748929
Corporate Tax Rates %                            0.322008
Schooling (years)                               -0.628293

                                Ease of Doing Business Rank  \
Women Entrepreneurship Index                      -0.713246
Entrepreneurship Index                            -0.717070
Inflation rate                                     0.466089
Female Labor Force Participation Rate             -0.360153
Gender Inequality Index                            0.748929
Ease of Doing Business Rank                        1.000000
Corporate Tax Rates %                              0.342884
Schooling (years)                                 -0.331999

                                Corporate Tax Rates %  \
Women Entrepreneurship Index                -0.199905
Entrepreneurship Index                      -0.166656
Inflation rate                               0.266044
Female Labor Force Participation Rate       -0.109179
Gender Inequality Index                      0.322008
Ease of Doing Business Rank                  0.342884
Corporate Tax Rates %                        1.000000
Schooling (years)                            0.016896

                                Schooling (years)
Women Entrepreneurship Index             0.721573
Entrepreneurship Index                   0.699228
Inflation rate                          -0.197663
Female Labor Force Participation Rate    0.122918
Gender Inequality Index                 -0.628293
Ease of Doing Business Rank             -0.331999
Corporate Tax Rates %                    0.016896
Schooling (years)                        1.000000
```

Preprocessing features before training the model with Column Transformer. Feature 'Gender Inequality Index' highly correlates with a few other features such as 'Entrepreneurship Index', 'Schooling (years)', and target variable 'Women Entrepreneurship Index'. Therefore, we had to drop it to account for multicollinearity.

In addition, we drop 'Country' since it does not have any impact on the model training.

[170]:
```
%load_ext autoreload
%autoreload 2
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, Normalizer, MinMaxScaler

#preprocessing features
preprocesser = ColumnTransformer([
    ('One-Hot Encoder', OneHotEncoder(), ['Level of development', 'European␣
 ↪Union Membership', 'Currency']),
    ('Drop', 'drop', ['Gender Inequality Index', 'Country']),
    ('Normalize', Normalizer(norm='l1'), ['Entrepreneurship Index', 'Inflation␣
 ↪rate',
                                          'Female Labor Force Participation␣
 ↪Rate',
                                          'Ease of Doing Business Rank',
                                          'Schooling (years)']),
    ('MinMax', MinMaxScaler(), ['Corporate Tax Rates %'])
])
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[171]:
```
#transforming features with Column Transformer
X_train_transformed = preprocesser.fit_transform(X_train)
X_val_transformed = preprocesser.fit_transform(X_val)
X_test_transformed = preprocesser.fit_transform(X_test)
```

Training and evaluating models

1. Stochastic Gradient Descent Regressor

[370]:
```
from sklearn.linear_model import SGDRegressor

lr = SGDRegressor(alpha=0.005, max_iter=2000) #min possible alpha

lr.fit(X_train_transformed, y_train)

y_pred_lr = lr.predict(X_train_transformed)
```

[173]:
```
from sklearn.metrics import mean_squared_error, r2_score, max_error

def print_regression_metrics(y, y_pred):
    print("MSE:", mean_squared_error(y, y_pred))
    print("R2 score:", r2_score(y, y_pred))
    print("Max error:", max_error(y, y_pred))
    print()
```

[371]:
```
print_regression_metrics(y_train, y_pred_lr)
```

```
MSE: 45.4648869531261
```

```
R2 score: 0.7889929941607101
Max error: 20.96293963529679
```

[175]:
```
# from sklearn.linear_model import SGDRegressor

# lr = SGDRegressor(alpha=1)
# lr.fit(X_train_transformed, y_train)
# y_pred_lr = lr.predict(X_train_transformed)
```

[176]:
```
# print_regression_metrics(y_train, y_pred_lr)
```

[177]:
```
# from sklearn.linear_model import SGDRegressor

# lr = SGDRegressor(alpha=0.5)

# lr.fit(X_train_transformed, y_train)
# y_pred_lr = lr.predict(X_train_transformed)
```

[178]:
```
# print_regression_metrics(y_train, y_pred_lr)
```

[179]:
```
# from sklearn.linear_model import SGDRegressor

# lr = SGDRegressor(alpha=5)
# lr.fit(X_train_transformed, y_train)
# y_pred_lr = lr.predict(X_train_transformed)
```

[180]:
```
# print_regression_metrics(y_train, y_pred_lr)
```

2. Lasso Regression Model

[334]:
```
from sklearn.linear_model import Lasso

reg = Lasso(alpha=0.003) #min possible alpha

%timeit reg.fit(X_train_transformed, y_train)
y_pred_lr = reg.predict(X_train_transformed)

import sys
import pickle

p = pickle.dumps(reg)
print(sys.getsizeof(p))
```

```
385 µs ± 9.94 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
661
```

[182]:
```
print_regression_metrics(y_train, y_pred_lr)
```

```
MSE: 37.58658716879849
R2 score: 0.8255569572540131
Max error: 17.65883081093734
```

[183]:
```
# from sklearn.linear_model import Lasso

# reg = Lasso(alpha=0.5) #min possible alpha

# reg.fit(X_train_transformed, y_train)
# y_pred_lr = reg.predict(X_train_transformed)
```

[184]:
```
# print_regression_metrics(y_train, y_pred_lr)
```

[185]:
```
# from sklearn.linear_model import Lasso

# reg = Lasso(alpha=1) #min possible alpha

# reg.fit(X_train_transformed, y_train)
# y_pred_lr = reg.predict(X_train_transformed)
```

[186]:
```
# print_regression_metrics(y_train, y_pred_lr)
```

[187]:
```
# from sklearn.linear_model import Lasso

# reg = Lasso(alpha=5) #min possible alpha

# reg.fit(X_train_transformed, y_train)
# y_pred_lr = reg.predict(X_train_transformed)
```

[188]:
```
# print_regression_metrics(y_train, y_pred_lr)
```

3. Linear Regression Model

[348]:
```
import sys
import pickle

lr2 = linear_model.LinearRegression()

%timeit  lr2.fit(X_train_transformed, y_train)
y_pred_lr = lr2.predict(X_train_transformed)


import sys
import pickle

p = pickle.dumps(lr2)
print(sys.getsizeof(p))
```

```
286 µs ± 8.15 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
672
```

[190]:
```
print_regression_metrics(y_train, y_pred_lr)
```

```
MSE: 35.30270512091824
R2 score: 0.8361566781575351
Max error: 18.25202381014924
```

[191]:
```
# #changing n_jobs did not affect results
# lr2 = linear_model.LinearRegression(n_jobs = 50)

# lr2.fit(X_train_transformed, y_train)
# y_pred_lr = lr2.predict(X_train_transformed)
```

[192]:
```
# print_regression_metrics(y_train, y_pred_lr)
```

4. Decision Tree Regressor

[193]:
```
# from sklearn.tree import DecisionTreeRegressor

# dt = DecisionTreeRegressor(max_depth=5)

# dt.fit(X_train_transformed, y_train)
# y_pred_dt = dt.predict(X_train_transformed)
```

[194]:
```
# print_regression_metrics(y_train, y_pred_dt)
```

[195]:
```
# from sklearn.tree import DecisionTreeRegressor

# dt = DecisionTreeRegressor(max_depth=6)

# dt.fit(X_train_transformed, y_train)
# y_pred_dt = dt.predict(X_train_transformed)
```

[196]:
```
# print_regression_metrics(y_train, y_pred_dt)
```

[349]:
```
import sys

from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor(max_depth=7)

%timeit dt.fit(X_train_transformed, y_train)
y_pred_dt = dt.predict(X_train_transformed)

p = pickle.dumps(dt)
print(sys.getsizeof(p))
```

```
269 µs ± 9.45 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
5694
```

[198]:
```
print_regression_metrics(y_train, y_pred_dt)
```

```
MSE: 0.008250000000000058
R2 score: 0.999961710939698
Max error: 0.25
```

[199]:
```
# from sklearn.tree import DecisionTreeRegressor

# dt = DecisionTreeRegressor(max_depth=8)

# dt.fit(X_train_transformed, y_train)
# y_pred_dt = dt.predict(X_train_transformed)
```

[200]:
```
# print_regression_metrics(y_train, y_pred_dt)
```

[201]:
```
# from sklearn.ensemble import VotingRegressor
# from sklearn.linear_model import LinearRegression
# from sklearn.ensemble import RandomForestRegressor

# r1 = LinearRegression()
# r2 = RandomForestRegressor(n_estimators=10, random_state=1)
# vr = VotingRegressor([('lr', r1), ('rf', r2)])
# vr.fit(X_train_transformed, y_train)
# y_pred_vr = vr.predict(X_train_transformed)
```

[202]:
```
# print_regression_metrics(y_train, y_pred_vr)
```

[203]:
```
# from sklearn.ensemble import VotingRegressor
# from sklearn.linear_model import LinearRegression
# from sklearn.ensemble import RandomForestRegressor

# r1 = LinearRegression()
# r2 = RandomForestRegressor(n_estimators=1000, random_state=1)
# vr = VotingRegressor([('lr', r1), ('rf', r2)])
# vr.fit(X_train_transformed, y_train)
# y_pred_vr = vr.predict(X_train_transformed)
```

[204]:
```
# print_regression_metrics(y_train, y_pred_vr)
```

[350]:
```
from sklearn.ensemble import VotingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

r1 = LinearRegression()
```

```
r2 = Lasso(alpha=0.003) #the best alpha parameter
vr = VotingRegressor([('lr', r1), ('ls', r2)])
%timeit vr.fit(X_train_transformed, y_train)
y_pred_vr = vr.predict(X_train_transformed)

p = pickle.dumps(vr)
print(sys.getsizeof(p))
```

```
1.15 ms ± 20.1 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
1380
```

[206]:
```
print_regression_metrics(y_train, y_pred_vr)
```

```
MSE: 35.87367563288831
R2 score: 0.8335067479316546
Max error: 17.955427310543286
```

[310]:
```
#Visualization
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


models = [['SGDRegressor', SGDRegressor()], ['Lasso', Lasso()], ['Linear␣
 ↪Regression',LinearRegression()],
          ['Voting Regressor', VotingRegressor([('lr', LinearRegression()),␣
 ↪('ls', Lasso())])]]
          #['Voting Regressor', VotingRegressor([('lr', r1), ('ls', r2)])]

#['Decision Tree', DecisionTreeRegressor(), 'Linear␣
 ↪Regression',LinearRegression()

model_score = []
r2_scores = []
mae_list = []
mse_list = []

for classifier in models:

    name = classifier[0]
    model = classifier[1]

    model.fit(X_train_transformed, pd.DataFrame(y_train).values.ravel())
    y_pred = model.predict(X_train_transformed)

    score = model.score(X_train_transformed, y_train)
    r2 = r2_score(y_train, y_pred)
    mae = mean_absolute_error(y_train, y_pred)
    mse = mean_squared_error(y_train, y_pred)
```

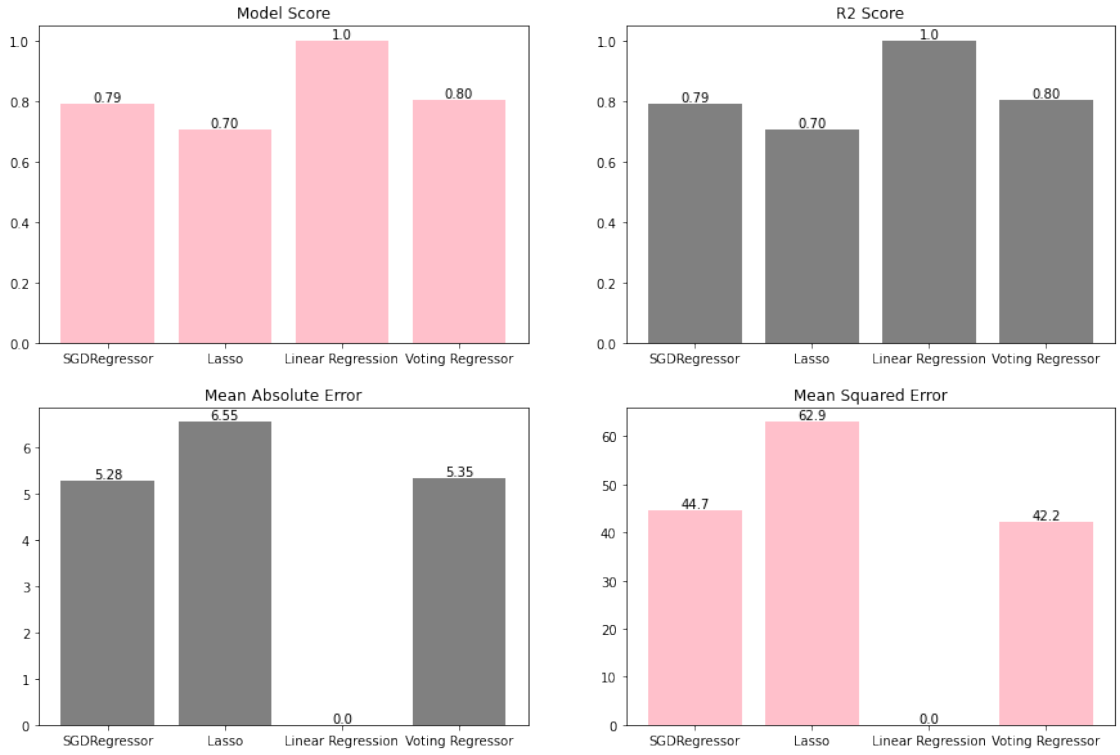
```
    model_score.append(score)
    r2_scores.append(r2)
    mae_list.append(mae)
    mse_list.append(mse)

    print(name)
    print('Model score:        ', score)
    print('R2 score:           ', r2)
    print('Mean absolute error:', mae)
    print('Mean squared error: ', mse)

    if model != models[-1][1]:
        print('')
```

```
SGDRegressor
Model score:         0.7924548155659654
R2 score:            0.7924548155659654
Mean absolute error: 5.277705145875703
Mean squared error:  44.7189812984024

Lasso
Model score:         0.7079255806616203
R2 score:            0.7079255806616203
Mean absolute error: 6.558030303030304
Mean squared error:  62.932178030303035

Linear Regression
Model score:         0.8361566781575351
R2 score:            0.8361566781575351
Mean absolute error: 4.790315800042106
Mean squared error:  35.30270512091824

Voting Regressor
Model score:         0.8040989037835562
R2 score:            0.8040989037835562
Mean absolute error: 5.358742736444603
Mean squared error:  42.21007334826447
```

```
C:\Users\regin\anaconda3\lib\site-
packages\sklearn\linear_model\_stochastic_gradient.py:1208: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
```

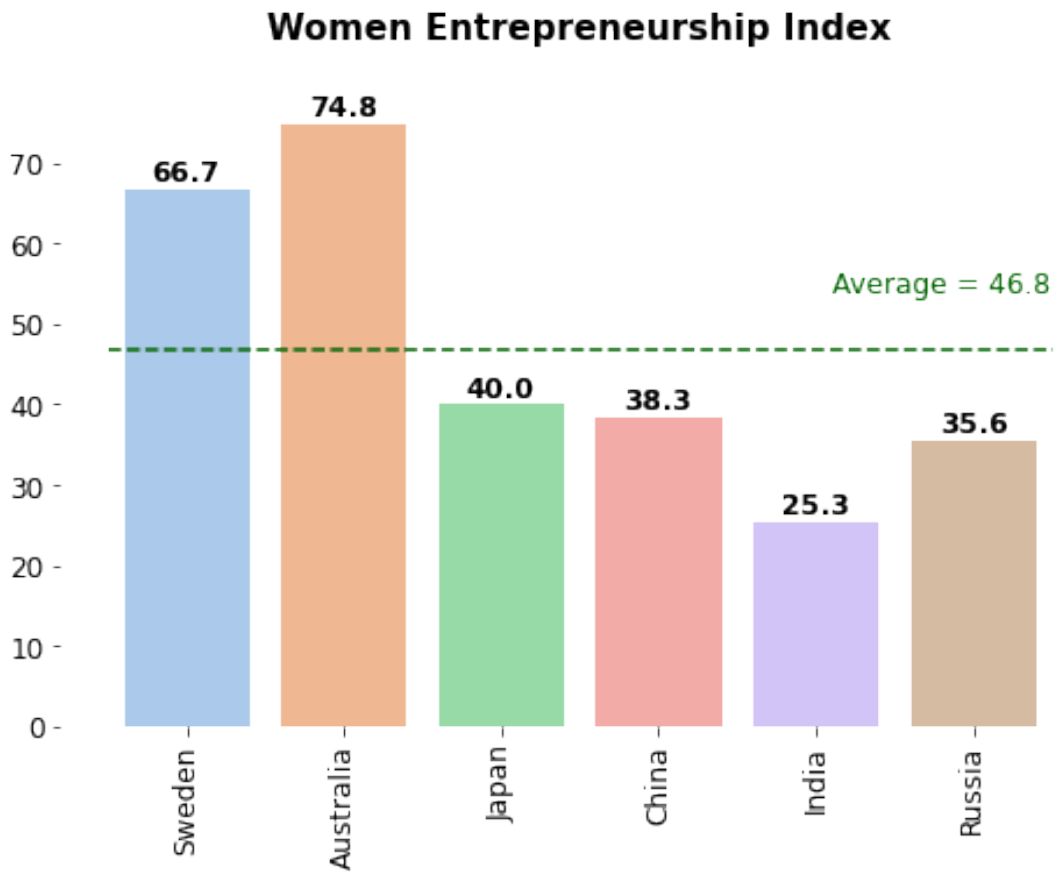
[309]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

scores = [model_score, r2_scores, mae_list, mse_list]
names = ['SGDRegressor', 'Lasso', 'Linear Regression', 'Voting Regressor']
score_names = ['Model Score', 'R2 Score', 'Mean Absolute Error', 'Mean Squared␣
 ↪Error']
colours = ['pink', 'gray', 'gray', 'pink']
i = 0

for row in axes:
    for ax in row:
        bars = ax.bar(names, scores[i], color=colours[i])

        for bar in bars:
            score = str(scores[i][bars.index(bar)])[:4]
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2., height, score,␣
 ↪ha='center', va='bottom')

        ax.set_title(score_names[i])
        i += 1

plt.show()
```



[374]:
```
import matplotlib.pyplot as plt

dfp = df[df['Country'].isin(['Russia','China', 'Australia', 'Sweden', 'India',␣
 ↪'Japan'])].copy()
#dfp.drop(columns=['No','Inflation rate'],inplace=True)
dfp.head()

#Creating the palette
palet = sns.color_palette(palette='pastel', n_colors=6)
#['#999999','#999999','#008000','#999999','#999999']

fig, ax = plt.subplots(1,1,figsize=(8,5))
by_country = sns.barplot(data=dfp, y='Women Entrepreneurship Index',␣
 ↪x='Country', palette=palet)

plt.ylabel('')
plt.xticks(fontsize=12, fontfamily='sans-serif', rotation=90)
plt.xlabel(' \n\n')
plt.title('Women Entrepreneurship Index\n', fontsize=15,␣
 ↪fontfamily='sans-serif', fontweight='bold')
fig.text(0.70,0.60,f'Average = {dfp["Women Entrepreneurship Index"].mean():.
 ↪1f}\n', fontweight='regular', color='darkgreen', fontfamily='sans-serif',␣
 ↪fontsize=12)
plt.yticks(fontsize=12)


ax.annotate(f'{float(dfp[dfp["Country"]=="Russia"]["Women Entrepreneurship␣
 ↪Index"].values)}',xy=(5,(float(dfp[dfp['Country']=='Russia']["Women␣
 ↪Entrepreneurship Index"].values)+1)), ha='center', fontsize=12,␣
 ↪fontfamily='sans-serif', fontweight = 'bold')
ax.annotate(f'{float(dfp[dfp["Country"]=="China"]["Women Entrepreneurship␣
 ↪Index"].values)}',xy=(3,(float(dfp[dfp['Country']=='China']["Women␣
 ↪Entrepreneurship Index"].values)+1)), ha='center', fontsize=12,␣
 ↪fontfamily='sans-serif', fontweight = 'bold')
ax.annotate(f'{float(dfp[dfp["Country"]=="Australia"]["Women Entrepreneurship␣
 ↪Index"].values)}',xy=(1,(float(dfp[dfp['Country']=='Australia']["Women␣
 ↪Entrepreneurship Index"].values)+1)), ha='center', fontsize=12,␣
 ↪fontfamily='sans-serif', fontweight = 'bold')
ax.annotate(f'{float(dfp[dfp["Country"]=="Sweden"]["Women Entrepreneurship␣
 ↪Index"].values)}',xy=(0,(float(dfp[dfp['Country']=='Sweden']["Women␣
 ↪Entrepreneurship Index"].values)+1)), ha='center', fontsize=12,␣
 ↪fontfamily='sans-serif', fontweight = 'bold')
ax.annotate(f'{float(dfp[dfp["Country"]=="India"]["Women Entrepreneurship␣
 ↪Index"].values)}',xy=(4,(float(dfp[dfp['Country']=='India']["Women␣
 ↪Entrepreneurship Index"].values)+1)), ha='center', fontsize=12,␣
 ↪fontfamily='sans-serif', fontweight = 'bold')
```

```
ax.annotate(f'{float(dfp[dfp["Country"]=="Japan"]["Women Entrepreneurship␣
 ↪Index"].values)}',xy=(2,(float(dfp[dfp['Country']=='Japan']["Women␣
 ↪Entrepreneurship Index"].values)+1)), ha='center', fontsize=12,␣
 ↪fontfamily='sans-serif', fontweight = 'bold')
plt.hlines(dfp["Women Entrepreneurship Index"].mean(),-0.5,5.5, color =␣
 ↪'darkgreen', label='Average', linestyles = 'dashed')

plt.box(False)
plt.show()
plt.savefig('wei.png')
```




```
<Figure size 432x288 with 0 Axes>
```

[158]:
```
#predicting on the test set
y_pred_lr = lr2.predict(X_test_transformed)
```

[159]:
```
print_regression_metrics(y_test, y_pred_lr)
```

```
MSE: 6.339462669548008
R2 score: 0.9085687390664086
Max error: 5.054636713840694
```

[318]:
```
y_pred_lr
```

[318]:
```
array([53.78869438, 45.6354948 , 45.75913318, 35.84620448, 63.81807362,
       27.75644183, 37.86264868, 68.02082464, 35.55264523, 53.35551907,
       60.37074561, 67.81128084, 63.77117064, 35.3944444 , 56.72222635,
       59.99795085, 62.37620869, 34.53203586, 30.33293806, 30.71964486,
       63.78178325, 30.9280982 , 29.78799956, 60.4973884 , 59.74576903,
       58.36730989, 33.97198478, 30.89439133, 62.59312328, 44.67956769,
       57.02035446, 58.25202381, 54.81237545, 34.81511505, 65.11568689,
       59.80999584, 36.78218456, 56.31236182, 30.73140315, 34.57675745])
```

[161]:
```
y_test
```

[161]:
```
31    31.1
43    36.9
30    31.6
13    55.9
46    37.0
40    41.2
Name: Women Entrepreneurship Index, dtype: float64
```

[ ]:
```

```