Lab 5 - Python

Table of contents

1 Data Analysis

1.1 Table Presentation

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

```
warnings.filterwarnings("ignore")

data = pd.read_csv(
    "https://raw.githubusercontent.com/alexanderquispe/CausalAI-Course/main/data/processed_es
)
```

```
data1 = (
    data.filter(
        [
            "y",
            "w",
            "gender_female",
            "gender_male",
            "gender_transgender",
            "age",
            "imd_decile",
        ]
    )
    .melt(
        id_vars=["y", "w", "age", "imd_decile"],
        value_vars=["gender_female", "gender_male", "gender_transgender"],
        var_name="gender",
        value_name="value",
    )
    .query("value > 0")
    .assign(
        gender=lambda df: df["gender"]
        .str.split("_")
        .str[-1]
        .map({"female": "Female", "male": "Male", "transgender": "Transgender"})
    )
    .drop(columns=["value"])
)

summary = (
    data1.groupby(["w", "gender"])
    .agg(
        n=("y", "size"),
        mean_y=("y", "mean"),
        sd_y=("y", "std"),
        mean_age=("age", "mean"),
        sd_age=("age", "std"),
```

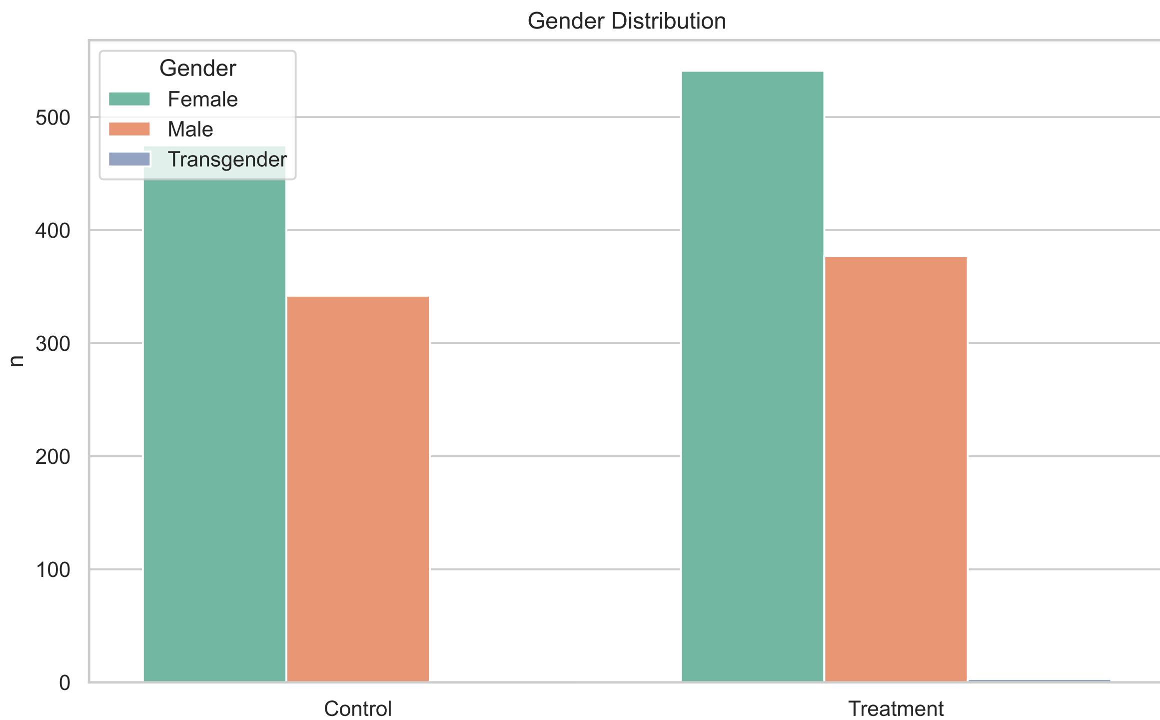
```
    )
    .reset_index()
    .assign(
        w=lambda df: df["w"].map({0: "Control", 1: "Treatment"}),
        gender=lambda df: df["gender"].map(
            {"Male": "Male", "Female": "Female", "Transgender": "Transgender"}
        ),
    )
    .rename(
        columns={
            "w": "Group",
            "gender": "Gender",
            "mean_y": "Mean - Y",
            "sd_y": "sd - Y",
            "mean_age": "Mean - Age",
            "sd_age": "sd - Age",
        }
    )
)

print(summary)
```

```
       Group       Gender    n  Mean - Y    sd - Y  Mean - Age  sd - Age
0    Control       Female  475  0.208421  0.406608   22.741053  3.591231
1    Control         Male  342  0.216374  0.412376   23.491228  3.547530
2    Control  Transgender    1  0.000000       NaN   17.000000       NaN
3  Treatment       Female  541  0.534196  0.499291   22.920518  3.500154
4  Treatment         Male  377  0.389920  0.488380   23.506631  3.575267
5  Treatment  Transgender    3  1.000000  0.000000   22.333333  3.214550
```

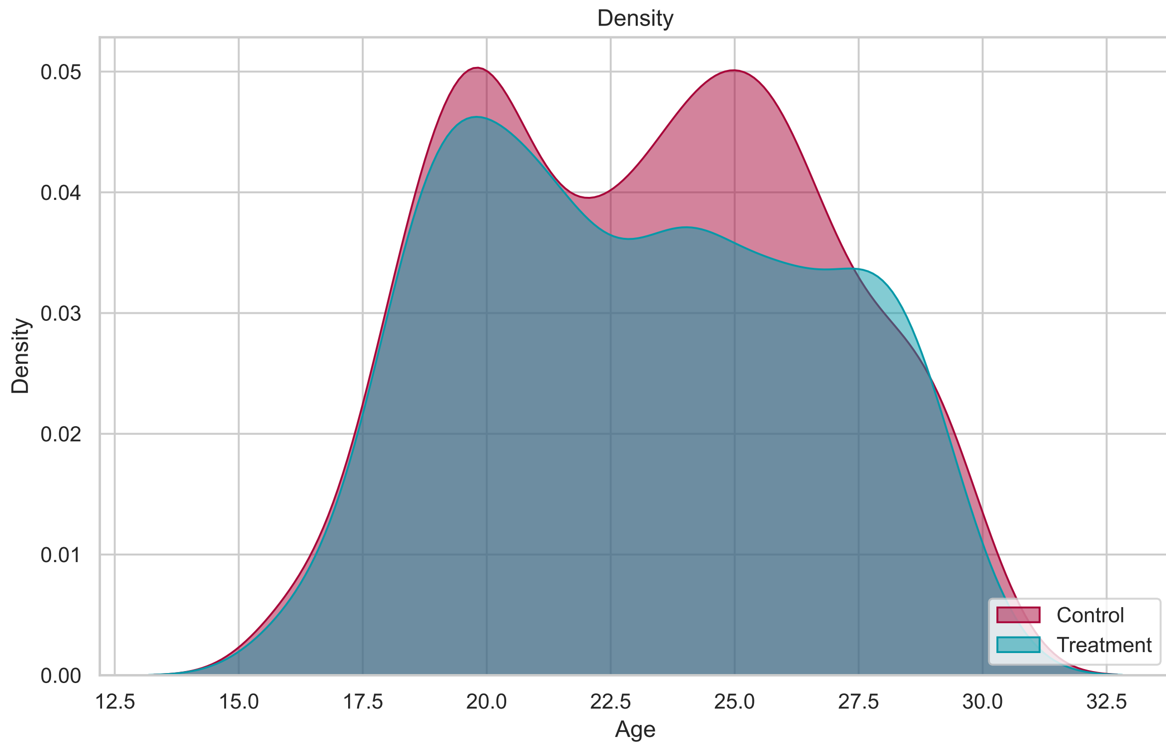
1.2 Graphs - Final output

```
sns.set_theme(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.countplot(data=data1, x='w', hue='gender', palette="Set2")
plt.xlabel('')
plt.ylabel('n')
plt.title('Gender Distribution')
plt.legend(title='Gender', loc='upper left')
```

```
plt.xticks([0, 1], ['Control', 'Treatment'])
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.kdeplot(data=data1, x='age', hue='w', fill=True, palette=['#0798a8', '#a8073a'], alpha=0
plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Density')
plt.legend(title='', loc='lower right', labels=['Control', 'Treatment'])
plt.show()
```



2 Linear Regression Analysis

2.1 LM 1

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
lm1 = smf.ols('y ~ w', data=data).fit()
lm1_summary = lm1.summary2().tables[1]
lm1_summary['model'] = "Simple"
lm1_summary
```

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]	model
Intercept	0.211491	0.016053	13.174492	7.533982e-38	0.180006	0.242977	Simple
w	0.265164	0.022059	12.020880	4.957428e-32	0.221900	0.308429	Simple

2.2 LM 2

```
base_formula = 'y ~ w + age + imd_decile'
lm2 = smf.ols(base_formula, data=data1).fit()
lm2_summary = lm2.summary2().tables[1]
lm2_summary['model'] = "Multiple"
lm2_summary
```

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]	model
Intercept	-0.147526	0.077502	-1.903520	5.713801e-02	-0.299533	0.004481	Multiple
w	0.263377	0.021907	12.022335	4.892918e-32	0.220409	0.306344	Multiple
age	0.015804	0.003069	5.149001	2.916669e-07	0.009784	0.021824	Multiple
imd_decile	-0.001501	0.007416	-0.202367	8.396539e-01	-0.016046	0.013045	Multiple

2.3 Double Lasso

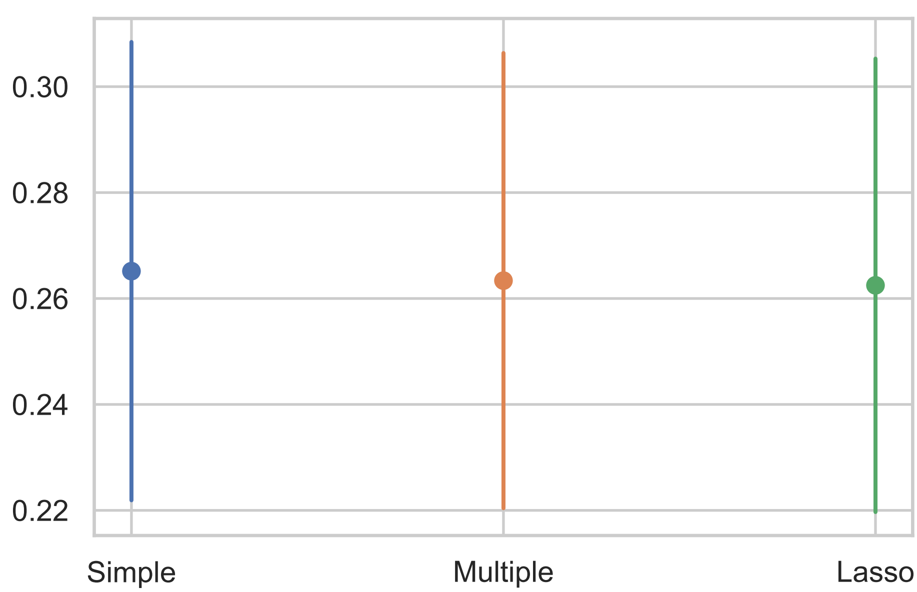
```
from sklearn.linear_model import LassoCV, Lasso
import numpy as np
cnt = [
    "w",
    "gender_female",
    "gender_male",
    "age",
    "imd_decile"
]

X = data[cnt]
y = data['y']

lasso_cv = LassoCV(alphas = None, cv = 4, max_iter=10).fit(X, y)
model_glm = Lasso(alpha = lasso_cv.alpha_).fit(X, y)
coefs = model_glm.coef_

relevant = np.nonzero(coefs)[0]

xvar= np.array(cnt)[relevant]
base_model = 'y ~ ' + " + ".join(xvar)
lm3 = smf.ols(base_model, data=data).fit()
lm3_summary = lm3.summary2().tables[1]
```

```python
lm3_summary['model'] = "Lasso"
lm3_summary
```

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]	model
Intercept	-0.133280	0.077287	-1.724486	8.479844e-02	-0.284866	0.018305	Lasso
w	0.262487	0.021823	12.028213	4.589079e-32	0.219685	0.305288	Lasso
gender_male	-0.085201	0.022225	-3.833634	1.307738e-04	-0.128791	-0.041611	Lasso
age	0.016887	0.003070	5.500006	4.363906e-08	0.010865	0.022909	Lasso
imd_decile	-0.002528	0.007392	-0.341942	7.324357e-01	-0.017026	0.011970	Lasso

2.4 Results

```python
combined_results = pd.concat([lm1_summary, lm2_summary, lm3_summary])
filtered_results = combined_results.query('index == "w"')
filtered_results.columns = ["coef", "std", "t", "p", "u", "l", "model"]
for model in filtered_results.model:
    ref = filtered_results.query("model == @model")
    x = [model, model]
    y = [ref["u"], ref["l"]]
    plt.scatter("model", "coef", data=ref)
    # plt.scatter('')
    plt.plot(x, y)
```



3 Non Linear Methods

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(
    data, stratify=data['y'], test_size=0.3, random_state=23
)
y_train = train['y']
d_train = train['w']
x_train = train.drop(columns = ['y', 'w'])

y_test = test['y']
d_test = test['w']
x_test = test.drop(columns = ['y', 'w'])
```

```
def lm_yd(y, d, md: str):
    df = pd.DataFrame({'y': y, 'd': d})
    model = smf.ols("y ~ d", data = df).fit()
    summary = model.summary2().tables[1]
    summary['model'] = md
    summary.columns = ['estimate', 'std.error', 't', 'p', 'u', 'l', 'model']
    return summary
```

3.1 Lasso

```
def residual_lasso(X_train, y_train, X_test, y_test, cv_n = 10):
    model = LassoCV(cv = cv_n).fit(X_test, y_test)
    residual = y_test - model.predict(X_test)
    return residual

y_r1 = residual_lasso(x_train, y_train, x_test, y_test)
d_r1 = residual_lasso(x_train, d_train, x_test, d_test)
l1 = lm_yd(y_r1, d_r1, "Lasso")
```

3.2 Regression Trees

```python
from sklearn.tree import DecisionTreeRegressor

def residual_dtree(X_train, y_train, X_test, y_test):
    model = DecisionTreeRegressor().fit(X_train, y_train)
    residual = y_test = model.predict(X_test)
    return residual

y_r2 = residual_dtree(x_train, y_train, x_test, y_test)
d_r2 = residual_dtree(x_train, d_train, x_test, d_test)

l2 = lm_yd(y_r2, d_r2, "Reg. Trees")
```

3.3 Boosting Trees

```python
from sklearn.ensemble import GradientBoostingRegressor

def gbm_residual(xtrain, xtest, ytrain, ytest):
    model = GradientBoostingRegressor().fit(xtrain, ytrain)
    residual = ytest - model.predict(xtest)
    return residual

y_r3 = gbm_residual(x_train, x_test, y_train, y_test)
d_r3 = gbm_residual(x_train, x_test, d_train, d_test)

l3 = lm_yd(y_r3, d_r3, "Bost. Trees")
```

3.4 Regresssion Forest

```python
from sklearn.ensemble import RandomForestRegressor

def rand_forest_res(x_train, y_train, x_test, y_test) :
    model = RandomForestRegressor().fit(x_train, y_train)
    residual = y_test - model.predict(x_test)
    return residual

y_r4 = rand_forest_res(x_train, y_train, x_test, y_test)
d_r4 = rand_forest_res(x_train, d_train, x_test, d_test)
```

```
l4 = lm_yd(y_r4, d_r4, "Reg. Forest")
```

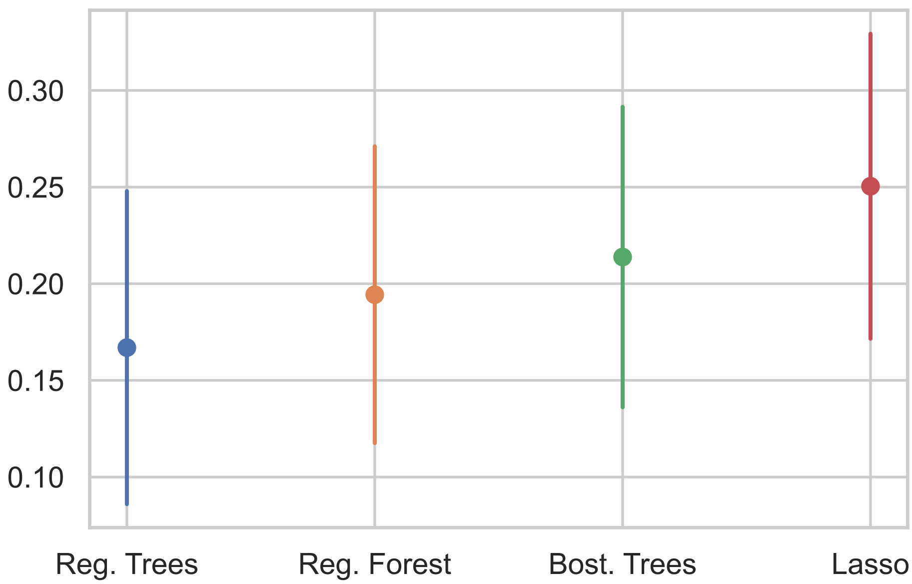
3.5 Results

```
from tabulate import tabulate

results = pd.concat([l1, l2, l3, l4])
results = results.query("index == 'd'").sort_values("estimate")

print(tabulate(results.round(2), headers="keys", tablefmt="pretty", showindex=False))
```

```
+----------+-----------+------+-----+------+------+-------------+
| estimate | std.error |  t   |  p  |  u   |  l   |    model    |
+----------+-----------+------+-----+------+------+-------------+
|   0.17   |   0.04    | 4.05 | 0.0 | 0.09 | 0.25 | Reg. Trees  |
|   0.19   |   0.04    | 4.97 | 0.0 | 0.12 | 0.27 | Reg. Forest |
|   0.21   |   0.04    | 5.4  | 0.0 | 0.14 | 0.29 | Bost. Trees |
|   0.25   |   0.04    | 6.24 | 0.0 | 0.17 | 0.33 |    Lasso    |
+----------+-----------+------+-----+------+------+-------------+
```

3.5.1 Plot

```
for md in results["model"]:
    ref = results.query("model == @md")
    plt.scatter("model", "estimate", data=ref)
    ci = ref[["u", "l"]].values[0]
    label = [md, md]
    plt.plot(label, ci)
```


0.30
0.25
0.20
0.15
0.10
Reg. Trees
Reg. Forest
Bost. Trees
Lasso