arsh_model_validation

July 22, 2021

[2]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.svm import SVC
```

0.1 FGNet-LOPO – Hierarchical Model

[179]:
```
df_fgnet = pd.read_csv("data/FGNet-LOPO.csv")
df_fgnet["ageclass"] = df_fgnet.age.apply(
    lambda r: 0 if r < 18 else 1
).astype(int)
df_fgnet['ageclass_ext'] = pd.cut(
    df_fgnet.age,
    bins=[0,13,16,20,24,28,32,36,40,100],
    labels=[0,1,2,3,4,5,6,7,8],
    include_lowest=True,
    ordered=False,
)

print(df_fgnet.columns)
print(f' <18: {df_fgnet[df_fgnet.age < 18].shape[0]}\n>=18: {df_fgnet[df_fgnet.
 ↪age >= 18].shape[0]}')
print(df_fgnet[["ageclass"]].value_counts())
print(df_fgnet[["ageclass_ext"]].value_counts())
df_fgnet[["age", "ageclass", "ageclass_ext"]].sample(20)
```

```
Index(['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'b10',
       …
       'b105', 'b106', 'b107', 'b108', 'b109', 'age', 'ID', 'Gender_0M_1F',
       'ageclass', 'ageclass_ext'],
      dtype='object', length=114)
 <18: 640
>=18: 362
ageclass
0           640
1           362
```

```
dtype: int64
ageclass_ext
0               513
2               118
1                99
3                64
8                60
4                51
5                38
6                36
7                23
dtype: int64
```

[179]:

	age	ageclass	ageclass_ext
300	3	0	0
444	0	0	0
895	5	0	0
692	2	0	0
640	2	0	0
765	3	0	0
976	12	0	0
646	11	0	0
575	18	1	2
709	16	0	1
260	30	1	5
830	12	0	0
704	4	0	0
122	10	0	0
598	4	0	0
232	12	0	0
251	3	0	0
564	16	0	1
299	0	0	0
359	5	0	0

[180]:
```python
def df_shape_table(*df_dicts):
    title_str = f"{'DataFrame':>15} | {'Shape':15}"
    print(title_str)
    for df_dict in df_dicts:
        print(f"{''.join(['-'] * 25):^33}")
        for name, df in df_dict.items():
            print(f"{name:>15} | { str(df.shape) :<15}")
```

[181]:
```python
df_yng = df_fgnet.iloc[df_fgnet[df_fgnet.ageclass == 0].index, :]
df_old = df_fgnet.iloc[df_fgnet[df_fgnet.ageclass == 1].index, :]

X = df_fgnet.drop(["age", "ID", "Gender_0M_1F", "ageclass"], axis=1)
X_yng = df_yng.drop(["age", "ID", "Gender_0M_1F", "ageclass"], axis=1)
```

```
X_old = df_old.drop(["age", "ID", "Gender_0M_1F", "ageclass"], axis=1)

y = df_fgnet[["ID", "age", "ageclass", "ageclass_ext", "Gender_0M_1F"]]
y_yng = df_yng[["ID", "age", "ageclass", "ageclass_ext", "Gender_0M_1F"]]
y_old = df_old[["ID", "age", "ageclass", "ageclass_ext", "Gender_0M_1F"]]

df_shape_table({'X': X, 'X Young': X_yng, 'X Old': X_old},
               {'y': y, 'y Young': y_yng, 'y Old': y_old})
```

```
  DataFrame | Shape
------------------------
          X | (1002, 110)
    X Young | (640, 110)
      X Old | (362, 110)
------------------------
          y | (1002, 5)
    y Young | (640, 5)
      y Old | (362, 5)
```

[182]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
X_yng_train, X_yng_test, y_yng_train, y_yng_test = train_test_split(X_yng,␣
↪y_yng, test_size=.2)
X_old_train, X_old_test, y_old_train, y_old_test = train_test_split(X_old,␣
↪y_old, test_size=.2)

df_shape_table(
    {'X_train': X_train, 'X_test': X_test, 'y_train': y_train, 'y_test':␣
↪y_test},
    {'X_yng_train': X_yng_train, 'X_yng_test': X_yng_test, 'y_yng_train':␣
↪y_yng_train, 'y_yng_test': y_yng_test},
    {'X_old_train': X_old_train, 'X_old_test': X_old_test, 'y_old_train':␣
↪y_old_train, 'y_old_test': y_old_test}
)
```

```
  DataFrame | Shape
------------------------
    X_train | (801, 110)
     X_test | (201, 110)
    y_train | (801, 5)
     y_test | (201, 5)
------------------------
X_yng_train | (512, 110)
 X_yng_test | (128, 110)
y_yng_train | (512, 5)
 y_yng_test | (128, 5)
------------------------
X_old_train | (289, 110)
 X_old_test | (73, 110)
```

```
y_old_train | (289, 5)
 y_old_test | (73, 5)
```

[ ]:
```python
def test_model(X, y, iter: int = 10, clf=None, reg=None):
    model_metrics = {}

    for i in range(10):
        model_metrics[i] = {
            "clf_true": [],
            "clf_pred": [],
            "reg_true": [],
            "reg_pred": [],
            "total": 0,
            "score": 0,
            "error": 0.
        }

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
        X_yng_train, _, y_yng_train, _ = train_test_split(X_yng, y_yng,␣
↪test_size=.2)
        X_old_train, _, y_old_train, _ = train_test_split(X_old, y_old,␣
↪test_size=.2)

        svm = SVC(C=1000, gamma = .01, kernel='rbf').fit(X_train,␣
↪y_train["ageclass"])
        reg_yng = Ridge(alpha=.1).fit(X_yng_train, y_yng_train["age"])
        reg_old = make_pipeline(StandardScaler(),
                                LinearSVR(epsilon=2.97, tol=1e-6, C=14.1)
                                ).fit(X_old_train, y_old_train["age"])

        for idx in X_test.index:
            row = X_test.loc[idx].values.reshape(1, -1)

            age_true = y_test.loc[idx, "age"]
            ageclass_true = y_test.loc[idx, "ageclass"]
            ageclass_pred = svm.predict(row)

            if ageclass_pred not in [0, 1]:
                raise ValueError("ageclass must be either 0 or 1")

            age_pred = reg_yng.predict(row) if ageclass_pred == 0 else reg_old.
↪predict(row)

            model_metrics[i]["error"] += abs(age_pred - age_true)
            model_metrics[i]["score"] += 1 if np.round(age_pred) == age_true␣
↪else 0
            model_metrics[i]["total"] += 1
```

```
            model_metrics[i]["clf_true"].append(ageclass_true)
            model_metrics[i]["clf_pred"].append(ageclass_pred)
            model_metrics[i]["reg_true"].append(age_true)
            model_metrics[i]["reg_pred"].append(age_pred)

        print(model_metrics[i]["error"] / model_metrics[i]["total"])

    df_clf2, df_reg2 = calculate_metrics(model_metrics)
```

[62]:
```
model_metrics = {}

for i in range(10):
    model_metrics[i] = {
        "clf_true": [],
        "clf_pred": [],
        "reg_true": [],
        "reg_pred": [],
        "total": 0,
        "score": 0,
        "error": 0.
    }

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
    X_yng_train, X_yng_test, y_yng_train, y_yng_test = train_test_split(X_yng,␣
↪y_yng, test_size=.2)
    X_old_train, X_old_test, y_old_train, y_old_test = train_test_split(X_old,␣
↪y_old, test_size=.2)

    svm = SVC(C=1000, gamma=.01, kernel='rbf').fit(X_train, y_train["ageclass"])
    reg_yng = Ridge(alpha=.1).fit(X_yng_train, y_yng_train["age"])
    reg_old = Ridge(alpha=.05).fit(X_old_train, y_old_train["age"])

    for idx in X_test.index:
        row = X_test.loc[idx].values.reshape(1, -1)

        age_true = y_test.loc[idx, "age"]
        ageclass_true = y_test.loc[idx, "ageclass"]
        ageclass_pred = svm.predict(row)

        if ageclass_pred not in [0, 1]:
            raise ValueError("ageclass must be either 0 or 1")

        age_pred = reg_yng.predict(row) if ageclass_pred == 0 else reg_old.
↪predict(row)

        model_metrics[i]["error"] += abs(age_pred - age_true)
        model_metrics[i]["score"] += 1 if np.round(age_pred) == age_true else 0
```

```
        model_metrics[i]["total"] += 1
        model_metrics[i]["clf_true"].append(ageclass_true)
        model_metrics[i]["clf_pred"].append(ageclass_pred)
        model_metrics[i]["reg_true"].append(age_true)
        model_metrics[i]["reg_pred"].append(age_pred)

    print(model_metrics[i]["error"] / model_metrics[i]["total"])
```

```
[3.47152899]
[3.77524155]
[3.8475331]
[3.81995785]
[3.36518051]
[3.2301345]
[3.81424444]
[3.66751503]
[3.77747745]
[3.79034929]
```

0.1.1 Classification/Regression Metrics

[196]:
```
from sklearn import metrics as m


def calculate_metrics(model_metrics, multi_class=False):
    clf_metrics = {
        'Accuracy': [],
        'Precision': [],
        'Recall': [],
        'F1 Score': [],
        'Jaccard': [],
    }

    reg_metrics = {
        "MAE": [],
        "MSE": [],
        "RMSE": [],
        "R2": [],
        "MAPE": [],
        "Median AE": [],
        "Max Error": [],
    }

    for _, metrics in model_metrics.items():
        clf_true, clf_pred = metrics["clf_true"], metrics["clf_pred"]
        reg_true, reg_pred = metrics["reg_true"], metrics["reg_pred"]
```

```
        if not multi_class:
            clf_metrics["Accuracy"].append(m.accuracy_score(clf_true, clf_pred))
            clf_metrics["Precision"].append(m.precision_score(clf_true,␣
↪clf_pred))
            clf_metrics["Recall"].append(m.recall_score(clf_true, clf_pred))
            clf_metrics["F1 Score"].append(m.f1_score(clf_true, clf_pred))
            clf_metrics["Jaccard"].append(m.jaccard_score(clf_true, clf_pred))

        else:
            clf_metrics["Accuracy"].append(m.accuracy_score(clf_true, clf_pred))
            clf_metrics["Precision"].append(m.precision_score(clf_true,␣
↪clf_pred, average=None))
            clf_metrics["Recall"].append(m.recall_score(clf_true, clf_pred,␣
↪average=None))
            clf_metrics["F1 Score"].append(m.f1_score(clf_true, clf_pred,␣
↪average=None))
            clf_metrics["Jaccard"].append(m.jaccard_score(clf_true, clf_pred,␣
↪average=None))

        reg_metrics["MAE"].append(m.mean_absolute_error(reg_true, reg_pred))
        reg_metrics["MSE"].append(m.mean_squared_error(reg_true, reg_pred))
        reg_metrics["RMSE"].append(m.mean_squared_error(reg_true, reg_pred,␣
↪squared=False))
        reg_metrics["R2"].append(m.r2_score(reg_true, reg_pred))
        reg_metrics["MAPE"].append(m.mean_absolute_percentage_error(reg_true,␣
↪reg_pred))
        reg_metrics["Median AE"].append(m.median_absolute_error(reg_true,␣
↪reg_pred))
        reg_metrics["Max Error"].append(m.max_error(reg_true, reg_pred))



    clf_metrics = pd.DataFrame(clf_metrics).round(4)
    reg_metrics = pd.DataFrame(reg_metrics).round(4)

    return clf_metrics, reg_metrics


clf_metrics, reg_metrics = calculate_metrics(model_metrics)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-196-f66d2e34d68b> in <module>
     55
     56
---> 57 clf_metrics, reg_metrics = calculate_metrics(model_metrics)
```

```
<ipython-input-196-f66d2e34d68b> in calculate_metrics(model_metrics, multi_clas
     27         if not multi_class:
     28             clf_metrics["Accuracy"].append(m.accuracy_score(clf_true,␣
 ↪clf_pred))
---> 29             clf_metrics["Precision"].append(m.precision_score(clf_true,
 ↪clf_pred))
     30             clf_metrics["Recall"].append(m.recall_score(clf_true,␣
 ↪clf_pred))
     31             clf_metrics["F1 Score"].append(m.f1_score(clf_true,␣
 ↪clf_pred))

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/utils/
 ↪validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/metrics
 ↪_classification.py in precision_score(y_true, y_pred, labels, pos_label,␣
 ↪average, sample_weight, zero_division)
   1651
   1652     """
-> 1653     p, _, _, _ = precision_recall_fscore_support(y_true, y_pred,
   1654                                                  labels=labels,
   1655                                                  pos_label=pos_label,

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/utils/
 ↪validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/metrics
 ↪_classification.py in precision_recall_fscore_support(y_true, y_pred, beta,␣
 ↪labels, pos_label, average, warn_for, sample_weight, zero_division)
   1459     if beta < 0:
   1460         raise ValueError("beta should be >=0 in the F-beta score")
-> 1461     labels = _check_set_wise_labels(y_true, y_pred, average, labels,
   1462                                     pos_label)
   1463
```

```
~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/metrics
↪_classification.py in _check_set_wise_labels(y_true, y_pred, average, labels,␣
↪pos_label)
   1289             if y_type == 'multiclass':
   1290                 average_options.remove('samples')
-> 1291             raise ValueError("Target is %s but average='binary'. Please "

   1292                              "choose another average setting, one of %r "
   1293                              % (y_type, average_options))

ValueError: Target is multiclass but average='binary'. Please choose another␣
↪average setting, one of [None, 'micro', 'macro', 'weighted'].
```

[83]:
```
reg_metrics
```

[83]:

	MAE	MSE	RMSE	R2	MAPE	Median AE	Max Error
0	3.4715	22.1504	4.7064	0.8765	4.989086e+14	2.5050	16.0771
1	3.7752	45.3650	6.7354	0.7734	3.619677e+14	2.2793	48.1820
2	3.8475	42.1088	6.4891	0.7348	2.913162e+14	2.0616	37.6786
3	3.8200	33.9145	5.8236	0.7874	2.341589e+14	2.1198	24.7566
4	3.3652	26.4973	5.1476	0.8341	4.635667e+14	1.7782	22.9836
5	3.2301	24.3302	4.9326	0.8488	2.577770e+14	1.8626	24.0144
6	3.8142	40.7614	6.3845	0.7392	4.190064e+14	2.0389	32.7234
7	3.6675	29.7576	5.4551	0.7927	4.145514e+14	2.2276	27.4794
8	3.7775	40.3120	6.3492	0.7865	4.791415e+14	2.1473	43.2820
9	3.7903	34.3981	5.8650	0.7710	4.435900e+14	2.2209	28.1663

[84]:
```
clf_metrics
```

[84]:

	Accuracy	Precision	Recall	F1 Score	Jaccard
0	0.8856	0.8750	0.8182	0.8456	0.7326
1	0.8607	0.7821	0.8472	0.8133	0.6854
2	0.8756	0.8667	0.8125	0.8387	0.7222
3	0.8557	0.7750	0.8493	0.8105	0.6813
4	0.8905	0.8312	0.8767	0.8533	0.7442
5	0.9154	0.9118	0.8493	0.8794	0.7848
6	0.8706	0.8060	0.8060	0.8060	0.6750
7	0.8905	0.8472	0.8472	0.8472	0.7349
8	0.8905	0.8507	0.8261	0.8382	0.7215
9	0.8259	0.7381	0.8267	0.7799	0.6392

0.2 FGNet-LOPO – Hierarchical Model w/ LOPOCV

[111]:
```
# from sklearn.preprocessing import LabelEncoder
# le = LabelEncoder()
# le.fit(df_fgnet.age)
```

[111]:
```
0       0
1       0
2       0
3       1
4       1
       ..
997     3
998     3
999     4
1000    4
1001    5
Name: age, Length: 1002, dtype: category
Categories (8, int64): [0 < 1 < 2 < 3 < 4 < 5 < 6 < 7]
```

[102]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVR, NuSVR
```

[103]:
```
model_metrics = {}

for i in range(10):
    model_metrics[i] = {
        "clf_true": [],
        "clf_pred": [],
        "reg_true": [],
        "reg_pred": [],
        "total": 0,
        "score": 0,
        "error": 0.
    }

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
    X_yng_train, _, y_yng_train, _ = train_test_split(X_yng, y_yng, test_size=.
↪2)
    X_old_train, _, y_old_train, _ = train_test_split(X_old, y_old, test_size=.
↪2)

    svm = SVC(C=1000, gamma = .01, kernel='rbf').fit(X_train,␣
↪y_train["ageclass"])
    reg_yng = Ridge(alpha=.1).fit(X_yng_train, y_yng_train["age"])
    reg_old = make_pipeline(StandardScaler(),
                            LinearSVR(epsilon=2.97, tol=1e-6, C=14.1)
                            ).fit(X_old_train, y_old_train["age"])

    for idx in X_test.index:
        row = X_test.loc[idx].values.reshape(1, -1)
```

```
        age_true = y_test.loc[idx, "age"]
        ageclass_true = y_test.loc[idx, "ageclass"]
        ageclass_pred = svm.predict(row)

        if ageclass_pred not in [0, 1]:
            raise ValueError("ageclass must be either 0 or 1")

        age_pred = reg_yng.predict(row) if ageclass_pred == 0 else reg_old.
↪predict(row)

        model_metrics[i]["error"] += abs(age_pred - age_true)
        model_metrics[i]["score"] += 1 if np.round(age_pred) == age_true else 0
        model_metrics[i]["total"] += 1
        model_metrics[i]["clf_true"].append(ageclass_true)
        model_metrics[i]["clf_pred"].append(ageclass_pred)
        model_metrics[i]["reg_true"].append(age_true)
        model_metrics[i]["reg_pred"].append(age_pred)

    print(model_metrics[i]["error"] / model_metrics[i]["total"])

df_clf2, df_reg2 = calculate_metrics(model_metrics)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-103-2a8795fdda05> in <module>
     19     reg_yng = Ridge(alpha=.1).fit(X_yng_train, y_yng_train["age"])
     20     reg_old = make_pipeline(StandardScaler(),
---> 21                             NuSVR(epsilon=2.97, tol=1e-6, C=14.1)
     22                             ).fit(X_old_train, y_old_train["age"])
     23

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/utils/
↪validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

TypeError: __init__() got an unexpected keyword argument 'epsilon'
```

[98]:
```
df_clf2, df_reg2 = calculate_metrics(model_metrics)
df_clf2
```

[98]:

	Accuracy	Precision	Recall	F1 Score	Jaccard
0	0.8657	0.7808	0.8382	0.8085	0.6786

```
1    0.8905     0.8588  0.8795    0.8690   0.7684
2    0.8856     0.8095  0.8226    0.8160   0.6892
3    0.8856     0.8400  0.8514    0.8456   0.7326
4    0.8657     0.8354  0.8250    0.8302   0.7097
5    0.8657     0.7841  0.8961    0.8364   0.7188
6    0.8458     0.8214  0.8118    0.8166   0.6900
7    0.8756     0.8667  0.8125    0.8387   0.7222
8    0.8607     0.8182  0.7714    0.7941   0.6585
9    0.8706     0.7857  0.8333    0.8088   0.6790
```

[99]:
```
df_reg2
```

[99]:

	MAE	MSE	RMSE	R2	MAPE	Median AE	Max Error
0	3.6252	33.3585	5.7757	0.8197	3.646839e+14	2.2161	30.0149
1	3.8108	35.6652	5.9720	0.8113	4.225795e+14	2.3778	28.3359
2	3.4905	31.2793	5.5928	0.8269	3.737717e+14	2.0753	27.5634
3	3.3936	24.7445	4.9744	0.8445	4.219817e+14	2.2171	20.8110
4	3.7128	34.2257	5.8503	0.7523	1.477490e+14	2.5456	32.5266
5	4.1854	46.6583	6.8307	0.7400	4.302189e+14	2.5074	28.9828
6	3.6364	36.3389	6.0282	0.7444	3.788209e+14	2.1914	35.8738
7	3.9131	37.3581	6.1121	0.7429	4.002109e+14	2.3741	26.9421
8	3.7520	33.8163	5.8152	0.7409	2.517691e+14	2.2373	24.8108
9	3.5871	34.0105	5.8319	0.8010	4.743715e+14	2.1886	25.4813

[101]:
```
df_reg2.describe().drop("count")
```

[101]:

	MAE	MSE	RMSE	R2	MAPE	Median AE	\
mean	3.710690	34.745530	5.878330	0.782390	3.666157e+14	2.293070	
std	0.224891	5.454431	0.460551	0.041952	9.676944e+13	0.151633	
min	3.393600	24.744500	4.974400	0.740000	1.477490e+14	2.075300	
25%	3.596625	33.472950	5.785575	0.743275	3.669559e+14	2.197575	
50%	3.674600	34.118100	5.841100	0.776650	3.895159e+14	2.227200	
75%	3.796100	36.170475	6.014150	0.817600	4.224301e+14	2.376875	
max	4.185400	46.658300	6.830700	0.844500	4.743715e+14	2.545600	

	Max Error
mean	28.134260
std	4.176009
min	20.811000
25%	25.846500
50%	27.949650
75%	29.756875
max	35.873800

0.3 Multilevel One-Hot

[183]:
```
y['ageclass_ext'].unique()
```

[183]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8]
Categories (9, int64): [0, 1, 2, 3, …, 5, 6, 7, 8]
```

[197]:
```
model_metrics = {}

for i in range(10):
    model_metrics[i] = {
        "clf_true": [],
        "clf_pred": [],
        "reg_true": [],
        "reg_pred": [],
        "total": 0,
        "score": 0,
        "error": 0.
    }

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)

    svm = SVC(C=1000, gamma=.01, kernel='rbf').fit(X_train,␣
↪y_train['ageclass_ext'])
    reg = [Ridge(alpha=.1),
           make_pipeline(StandardScaler(),
                         LinearSVR(epsilon=2.97, tol=1e-6, C=14.1)),
           Ridge(alpha=.1), Ridge(alpha=.1), Ridge(alpha=.1),
           Ridge(alpha=.1), Ridge(alpha=.1), Ridge(alpha=.1),
           Ridge(alpha=.1)]

    for r in range(len(reg)):
        indices = y_train[y_train['ageclass_ext'] == r].index
        reg[r].fit(X_train.loc[indices], y_train.loc[indices,'age'])

    # reg_yng = Ridge(alpha=.1).fit(X_yng_train, y_yng_train["age"])
    # reg_old = make_pipeline(StandardScaler(),
    #                         LinearSVR(epsilon=2.97, tol=1e-6, C=14.1)
    #                         ).fit(X_old_train, y_old_train["age"])

    for idx in X_test.index:
        row = X_test.loc[idx].values.reshape(1, -1)
        ageclass_true = y_test.loc[idx, 'ageclass_ext']
        ageclass_pred = svm.predict(row)

        # if ageclass_pred not in list(y['ageclass_ext'].unique()):
        #     raise ValueError("ageclass must be in {}".
↪format(str(y['ageclass_ext'].unique())))
```

```
        age_pred = reg[ageclass_pred[0]].predict(row)
        # age_pred = reg_yng.predict(row) if ageclass_pred == 0 else reg_old.
↪predict(row)

        model_metrics[i]["error"] += abs(age_pred - age_true)
        model_metrics[i]["score"] += 1 if np.round(age_pred) == age_true else 0
        model_metrics[i]["total"] += 1
        model_metrics[i]["clf_true"].append(ageclass_true)
        model_metrics[i]["clf_pred"].append(ageclass_pred)
        model_metrics[i]["reg_true"].append(age_true)
        model_metrics[i]["reg_pred"].append(age_pred)

    print(model_metrics[i]["error"] / model_metrics[i]["total"])

df_clf2, df_reg2 = calculate_metrics(model_metrics, multi_class=True)
```

```
[13.03282024]
[13.09861697]
[12.8630153]
[13.56481272]
[12.84421362]
[13.84267491]
[12.63527465]
[13.7707184]
[13.27748506]
[12.572591]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-197-672e35cb270c> in <module>
     52     print(model_metrics[i]["error"] / model_metrics[i]["total"])
     53
---> 54 df_clf2, df_reg2 = calculate_metrics(model_metrics)

<ipython-input-196-f66d2e34d68b> in calculate_metrics(model_metrics, multi_clas
     27         if not multi_class:
     28             clf_metrics["Accuracy"].append(m.accuracy_score(clf_true,␣
 ↪clf_pred))
---> 29             clf_metrics["Precision"].append(m.precision_score(clf_true,
 ↪clf_pred))
     30             clf_metrics["Recall"].append(m.recall_score(clf_true,␣
 ↪clf_pred))
     31             clf_metrics["F1 Score"].append(m.f1_score(clf_true,␣
 ↪clf_pred))

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/utils/
 ↪validation.py in inner_f(*args, **kwargs)
```

```
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/metrics
 ↪_classification.py in precision_score(y_true, y_pred, labels, pos_label,␣
 ↪average, sample_weight, zero_division)
   1651
   1652     """
-> 1653     p, _, _, _ = precision_recall_fscore_support(y_true, y_pred,
   1654                                                  labels=labels,
   1655                                                  pos_label=pos_label,

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/utils/
 ↪validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/metrics
 ↪_classification.py in precision_recall_fscore_support(y_true, y_pred, beta,␣
 ↪labels, pos_label, average, warn_for, sample_weight, zero_division)
   1459     if beta < 0:
   1460         raise ValueError("beta should be >=0 in the F-beta score")
-> 1461     labels = _check_set_wise_labels(y_true, y_pred, average, labels,
   1462                                     pos_label)
   1463

~/.pyenv/versions/anaconda3-2021.05/lib/python3.8/site-packages/sklearn/metrics
 ↪_classification.py in _check_set_wise_labels(y_true, y_pred, average, labels,
 ↪pos_label)
   1289             if y_type == 'multiclass':
   1290                 average_options.remove('samples')
-> 1291             raise ValueError("Target is %s but average='binary'. Please "
   1292                              "choose another average setting, one of %r "
   1293                              % (y_type, average_options))

ValueError: Target is multiclass but average='binary'. Please choose another␣
 ↪average setting, one of [None, 'micro', 'macro', 'weighted'].
```

[ ]: