Assignment5

July 4, 2024

1 Assignment 5

1.1 1. Choose a REGRESSION dataset (reusing bikeshare is allowed), perform a test/train split, and build a regression model (just like in assignment 3), and calculate the

```
+ Training Error (MSE, MAE)
+ Testing Error (MSE, MAE)
```

1.2 2. Choose a CLASSIFICATION dataset (not the adult.data set, The UCI repository has many datasets as well as Kaggle), perform test/train split and create a classification model (your choice but DecisionTree is fine). Calculate

```
+ Accuracy
+ Confusion Matrix
+ Classifcation Report
```

1.3 3. (Bonus) See if you can improve the classification model's performance with any tricks you can think of (modify features, remove features, polynomial features)

[3]:

```
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import linear_model, metrics
plt.rcParams['figure.figsize'] = 20, 10
import pandas as pd
import numpy as np

bikeshare_data = pd.read_csv('bikeshare_hour_count.csv')
bikeshare_data.dropna(inplace=True)
bikeshare_data
```

[3]:

```
    hour  monday  tuesday  wednesday  thursday  friday  saturday  sunday
0    0.0    21.0     34.0       43.0      47.0    51.0      89.0   106.0
1    0.1    39.0     22.0       27.0      37.0    56.0      87.0   100.0
2    0.2    31.0     24.0       26.0      42.0    50.0      98.0    77.0
3    0.3    26.0     27.0       25.0      29.0    52.0      99.0    87.0
```

```
4     0.4    19.0     24.0       29.0      29.0    50.0      98.0    69.0
..     …      …       …       …       …      …       …      …
235  23.5    36.0     65.0       60.0      94.0    80.0      93.0    28.0
236  23.6    37.0     61.0       66.0     100.0    81.0      95.0    28.0
237  23.7    30.0     42.0       49.0      80.0   101.0     105.0    27.0
238  23.8    33.0     52.0       47.0      79.0    91.0      93.0    24.0
239  23.9    34.0     33.0       48.0      65.0   105.0     111.0    23.0

[235 rows x 8 columns]
```

```
[4]: #Part 1
     from sklearn.linear_model import LinearRegression
     from sklearn import linear_model

     x = bikeshare_data['hour'].values.reshape(-1,1)
     y = bikeshare_data['saturday'].values.reshape(-1,1)

     from sklearn.model_selection import train_test_split
     xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2)
     linear = linear_model.LinearRegression().fit(xtrain, ytrain)
```

```
[6]: (
         metrics.mean_squared_error(ytest, linear.predict(xtest))
     )
```

```
[6]: 23559.85808987989
```

```
[8]: (
         metrics.mean_absolute_error(ytest, linear.predict(xtest))
     )
```

```
[8]: 136.0879897318393
```

```
[4]: #Part 2
     heart_data = pd.read_csv('heart.csv')
     heart_data_golden = pd.read_csv('heart.csv')
     heart_data.dropna()
     heart_data.dropna()
     heart_data
```

```
[4]:       age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
     0      52    1   0       125   212    0        1      168      0      1.0
     1      53    1   0       140   203    1        0      155      1      3.1
     2      70    1   0       145   174    0        1      125      1      2.6
     3      61    1   0       148   203    0        1      161      0      0.0
     4      62    0   0       138   294    1        1      106      0      1.9
     …   …   …   ..       …   …   …      …      …     …      …
```

```
1020    59    1   1       140   221    0        1       164      1       0.0
1021    60    1   0       125   258    0        0       141      1       2.8
1022    47    1   0       110   275    0        0       118      1       1.0
1023    50    0   0       110   254    0        0       159      0       0.0
1024    54    1   0       120   188    0        1       113      0       1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
…       …  ..   …     …
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]
```

```
[5]: heart_data['oldpeak'].unique()
```

```
[5]: array([1. , 3.1, 2.6, 0. , 1.9, 4.4, 0.8, 3.2, 1.6, 3. , 0.7, 4.2, 1.5,
            2.2, 1.1, 0.3, 0.4, 0.6, 3.4, 2.8, 1.2, 2.9, 3.6, 1.4, 0.2, 2. ,
            5.6, 0.9, 1.8, 6.2, 4. , 2.5, 0.5, 0.1, 2.1, 2.4, 3.8, 2.3, 1.3,
            3.5])
```

```
[6]: non_numeric_columns = ['sex','cp','fbs','restecg','exang', 'thal']
```

```
[7]: x = heart_data.copy().drop(non_numeric_columns, axis=1)
```

```
[8]: xt = heart_data_golden.copy().drop(non_numeric_columns, axis=1)
```

```
[9]: x['oldpeak'] = x.oldpeak.str.contains('.').astype(int)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[9], line 1
----> 1 x['oldpeak'] = x.oldpeak.str.contains('.').astype(int)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/generic.py:5989, in NDFrame.__getattr__(self, name)
   5982 if (
   5983     name not in self._internal_names_set
   5984     and name not in self._metadata
   5985     and name not in self._accessors
   5986     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5987 ):
   5988     return self[name]
-> 5989 return object.__getattribute__(self, name)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/accessor.py:224, in CachedAccessor.__get__(self, obj, cls)
    221 if obj is None:
    222     # we're accessing the attribute of the class, i.e., Dataset.geo
    223     return self._accessor
--> 224 accessor_obj = self._accessor(obj)
    225 # Replace the property with the accessor object. Inspired by:
    226 # https://www.pydanny.com/cached-property.html
    227 # We need to use object.__setattr__ because we overwrite __setattr__ on
    228 # NDFrame
    229 object.__setattr__(obj, self._name, accessor_obj)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/strings/accessor.py:181, in StringMethods.__init__(self, data)
    178 def __init__(self, data) -> None:
    179     from pandas.core.arrays.string_ import StringDtype
--> 181     self._inferred_dtype = self._validate(data)
    182     self._is_categorical = is_categorical_dtype(data.dtype)
    183     self._is_string = isinstance(data.dtype, StringDtype)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/strings/accessor.py:235, in StringMethods._validate(data)
    232 inferred_dtype = lib.infer_dtype(values, skipna=True)
    234 if inferred_dtype not in allowed_types:
--> 235     raise AttributeError("Can only use .str accessor with string values
 ↪")
    236 return inferred_dtype

AttributeError: Can only use .str accessor with string values!
```

[51]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

[53]:
```
model = DecisionTreeClassifier(criterion='entropy')
```

[55]:
```
model.fit(x.drop(['oldpeak'], axis=1), x.oldpeak)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[55], line 1
----> 1 model.fit(x.drop(['species'], axis=1), x.species)
```

```
File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/frame.py:5258, in DataFrame.drop(self, labels, axis, index,␣
 ↪columns, level, inplace, errors)
   5110 def drop(
   5111     self,
   5112     labels: IndexLabel = None,
   (…)
   5119     errors: IgnoreRaise = "raise",
   5120 ) -> DataFrame | None:
   5121     """
   5122     Drop specified labels from rows or columns.
   5123 
   (…)
   5256             weight  1.0     0.8
   5257     """
-> 5258     return super().drop(
   5259         labels=labels,
   5260         axis=axis,
   5261         index=index,
   5262         columns=columns,
   5263         level=level,
   5264         inplace=inplace,
   5265         errors=errors,
   5266     )

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/generic.py:4549, in NDFrame.drop(self, labels, axis, index,␣
 ↪columns, level, inplace, errors)
   4547 for axis, labels in axes.items():
   4548     if labels is not None:
-> 4549         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4551 if inplace:
   4552     self._update_inplace(obj)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/generic.py:4591, in NDFrame._drop_axis(self, labels, axis, level,␣
 ↪errors, only_slice)
   4589         new_axis = axis.drop(labels, level=level, errors=errors)
   4590     else:
-> 4591         new_axis = axis.drop(labels, errors=errors)
   4592     indexer = axis.get_indexer(new_axis)
   4594 # Case for non-unique axis
   4595 else:

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/indexes/base.py:6699, in Index.drop(self, labels, errors)
   6697 if mask.any():
   6698     if errors != "ignore":
```

```
-> 6699         raise KeyError(f"{list(labels[mask])} not found in axis")
   6700     indexer = indexer[~mask]
   6701 return self.delete(indexer)

KeyError: "['species'] not found in axis"
```

```
[ ]: list(zip(x.drop(['oldpeak'], axis=1).columns, model.feature_importances_))
```

```
[ ]: x.drop(['oldpeak'] , axis=1).head()
```

```
[ ]: set(x.columns) - set(xt.columns)
```

```
[ ]: predictions = model.predict(xt.drop(['oldpeak'], axis=1))
```

```
[ ]: predictions_train = model.predict(x.drop(['oldpeak'], axis=1))
```

```
[ ]: xt.drop(['oldpeak'], axis=1).head()
```

```
[ ]: # References
     Janosi,Andras, Steinbrunn, William, Pfisterer, Matthias,
     and Detrano,Robert.
     (1988).
     Heart Disease.
     UCI Machine Learning Repository.
     https://doi.org/10.24432/C52P4X.
```