Prediction of “or” occurrences based on “and” occurrences

1 Introduction

The words “and” and “or” are commonly used in books. The question at hand is, what is the relationship between these two words. A potential hypothesis might be that if a lot of “and” are used then the literary level of the work might be lower and thus contain more occurrences of the word “or”.

2 Problem Formulation

The datapoints are books. The books are chosen from the site gutenberg.org which hosts 60000 free e-books (this might not be a very representative sample of all books). The label is the number of occurrences of the word “or” in the book’s text body (integer). The feature is the number of occurrences of the word “and” (integer).

3 Methods

There are 120 datapoints after the data was cleaned (by dropping duplicate, rows where the occurrences were somehow 0, and rows where the occurrences of “and” were above 6000 since there weren’t a lot of those rows). The method used is polynomial regression since there is only one feature which couldn’t on its own provide a good enough predictor. The feature was chosen for its simplicity. The feature is transformed into a vector of polynomial powers of that feature and then used as the input for a linear regressor. The choice for the loss function was the mean squared error loss because its a good default choice for a loss function since it was favourable properties (convex, differentiable). The data was split into training, validation, test sets in a 0.60, 0.30, 0.10 split. There isn’t an easy way to find an optimal ratio so this will just do.

number of ”or” occurrences

March 10, 2022

[6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

#scraped, cleaned and stored data
dataframe = pd.read_pickle("bookData.pkl")

#extract feature matrix and label vector
X = dataframe['Count of "and"'].to_numpy().reshape(-1, 1) #feature: number of␣
↪occurances of the word "and"
y = dataframe['Count of "or"'].to_numpy() #label: number of occurances of the␣
↪word "or"

#Polynomial regression:

degrees = [1, 2, 3, 4]
tr_errors, val_errors, test_errors = [], [], []
fig, axs = plt.subplots(len(degrees)+1, 1)

for i, degree in enumerate(degrees):
    #create polynomial deatures object
    poly = PolynomialFeatures(degree = degree)

    #split the data into training, validation and test sets
    X_train, X_inter, y_train, y_inter = train_test_split(X, y, test_size = 0.
↪4, random_state = 100)
    X_val, X_test, y_val, y_test = train_test_split(X, y, test_size = 0.25,␣
↪random_state = 100)

    #transform feeatures
    X_train_poly = poly.fit_transform(X_train, y_train)
    X_val_poly = poly.fit_transform(X_val, y_val)
    X_test_poly = poly.fit_transform(X_test, y_test)
    #create and train linear regressor
    lin_regr = LinearRegression(fit_intercept=False)
    lin_regr.fit(X_train_poly, y_train)

    #calculate training error
    y_train_pred = lin_regr.predict(X_train_poly)
    tr_errors.append(mean_squared_error(y_train, y_train_pred))

    #calculate validation error
    y_val_pred = lin_regr.predict(X_val_poly)
    val_errors.append(mean_squared_error(y_val, y_val_pred))

    #calculate test error for later
    y_pred_test = lin_regr.predict(X_test_poly)
    test_errors.append(mean_squared_error(y_test, y_pred_test))

    #plot hypothesis
    X_fit = np.linspace(X.min(), X.max(), 100)
    axs[i].plot(X_fit, lin_regr.predict(poly.transform(X_fit.reshape(-1, 1))),␣
 ↪label="Model" )
    axs[i].scatter(X, y, color='b', s=10)
    axs[i].set_ylabel("label")
    axs[i].set_xlabel("feature")

#choose and print best model (polynomial degree) and print the test error for␣
 ↪that
min_val_index = val_errors.index(min(val_errors))
print(degrees[min_val_index])
print(test_errors[min_val_index])

#plot validation and training error for different degrees
axs[-1].plot(degrees, tr_errors, label="training error")
axs[-1].plot(degrees, val_errors, label="validation error")
axs[-1].set_ylabel("error")
axs[-1].set_xlabel("degree")
axs[-1].legend()

plt.show()
```

```
            ␣
↪---------------------------------------------------------------------------

       AttributeError                            Traceback (most recent call␣
↪last)
```

```
    <ipython-input-6-334e2a9329df> in <module>
      8
      9 #scraped, cleaned and stored data
---> 10 dataframe = pd.read_pickle("bookData.pkl")
     11
     12 #extract feature matrix and label vector


    /opt/conda/lib/python3.8/site-packages/pandas/io/pickle.py in␣
↪read_pickle(filepath_or_buffer, compression)
    185             #  "No module named 'pandas.core.sparse.series'"
    186             #  "Can't get attribute '__nat_unpickle' on <module␣
↪'pandas._libs.tslib"
--> 187             return pc.load(f, encoding=None)
    188     except UnicodeDecodeError:
    189         # e.g. can occur for files written in py27; see GH#28645 and␣
↪GH#31988


    /opt/conda/lib/python3.8/site-packages/pandas/compat/pickle_compat.py in␣
↪load(fh, encoding, is_verbose)
    247         up.is_verbose = is_verbose
    248
--> 249         return up.load()
    250     except (ValueError, TypeError):
    251         raise


    /opt/conda/lib/python3.8/pickle.py in load(self)
   1208                     raise EOFError
   1209                 assert isinstance(key, bytes_types)
-> 1210                 dispatch[key[0]](self)
   1211         except _Stop as stopinst:
   1212             return stopinst.value


    /opt/conda/lib/python3.8/pickle.py in load_stack_global(self)
   1533         if type(name) is not str or type(module) is not str:
   1534             raise UnpicklingError("STACK_GLOBAL requires str")
-> 1535         self.append(self.find_class(module, name))
   1536     dispatch[STACK_GLOBAL[0]] = load_stack_global
   1537


    /opt/conda/lib/python3.8/site-packages/pandas/compat/pickle_compat.py in␣
↪find_class(self, module, name)
    187         key = (module, name)
    188         module, name = _class_locations_map.get(key, key)
--> 189         return super().find_class(module, name)
    190
    191


    /opt/conda/lib/python3.8/pickle.py in find_class(self, module, name)
   1577         __import__(module, level=0)
   1578         if self.proto >= 4:
-> 1579             return _getattribute(sys.modules[module], name)[0]
   1580         else:
   1581             return getattr(sys.modules[module], name)


    /opt/conda/lib/python3.8/pickle.py in _getattribute(obj, name)
    329             obj = getattr(obj, subpath)
    330         except AttributeError:
--> 331             raise AttributeError("Can't get attribute {!r} on {!r}"
    332                                  .format(name, obj)) from None
    333     return obj, parent


    AttributeError: Can't get attribute 'new_block' on <module 'pandas.core.
↪internals.blocks' from '/opt/conda/lib/python3.8/site-packages/pandas/core/
↪internals/blocks.py'>
```

[ ]: