Protein classification

June 17, 2020

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.preprocessing import OneHotEncoder
```

```
In [2]: data=pd.read_csv("/home/bscuser/data/train.csv")
```

```
In [3]: data
```

```
Out[3]:        Sequence  Active
        0          DKWL       0
        1          FCHN       0
        2          KDQP       0
        3          FNWI       0
        4          NKRM       0
        ...         ...     ...
        111995     GSME       0
        111996     DLPT       0
        111997     SGHC       0
        111998     KIGT       0
        111999     PGPT       0

        [112000 rows x 2 columns]
```

```
In [5]: def split(word):
            return [char for char in word]
```

```
In [14]: first=[]
         second=[]
         third=[]
         fourth=[]
         for index,row in data.iterrows():
             first.append(split(row[0])[0])
             second.append(split(row[0])[1])
             third.append(split(row[0])[2])
             fourth.append(split(row[0])[3])
```

```
In [21]: data_split=pd.DataFrame()
         data_split["First"]=first
         data_split["Second"]=second
         data_split["Third"]=third
         data_split["Fourth"]=fourth
         enc = OneHotEncoder(handle_unknown='ignore')
         enc.fit(data_split)
```

```
Out[21]: OneHotEncoder(categories='auto', drop=None, dtype=<class 'numpy.float64'>,
                       handle_unknown='ignore', sparse=True)
```

```
In [22]: enc.categories_
```

```
Out[22]: [array(['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P',
                 'Q', 'R', 'S', 'T', 'V', 'W', 'Y'], dtype=object),
          array(['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P',
                 'Q', 'R', 'S', 'T', 'V', 'W', 'Y'], dtype=object),
          array(['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P',
                 'Q', 'R', 'S', 'T', 'V', 'W', 'Y'], dtype=object),
          array(['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P',
                 'Q', 'R', 'S', 'T', 'V', 'W', 'Y'], dtype=object)]
```

```
In [23]: enc_data=enc.transform(data_split).toarray()
```

```
In [25]: import numpy as np
         from sklearn.model_selection import train_test_split
         X=enc_data
         y=np.array(data["Active"])
         X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1, random_state=4
```

```
In [26]: from sklearn.pipeline import make_pipeline
         from sklearn.preprocessing import StandardScaler
         from sklearn.svm import SVC
```

```
In [27]: clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
         clf.fit(X_train, y_train)
```

```
Out[27]: Pipeline(memory=None,
                  steps=[('standardscaler',
                          StandardScaler(copy=True, with_mean=True, with_std=True)),
                         ('svc',
                          SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None,
                              coef0=0.0, decision_function_shape='ovr', degree=3,
                              gamma='auto', kernel='rbf', max_iter=-1, probability=False,
                              random_state=None, shrinking=True, tol=0.001,
                              verbose=False))],
                  verbose=False)
```

```
In [28]: from sklearn.metrics import f1_score
         y_pred=clf.predict(X_val)
         f1_score(y_val, y_pred, average='macro')
```

```
Out[28]: 0.917703465156569
```

```
In [30]: test_data=pd.read_csv("/home/bscuser/data/test.csv")
```

```
In [31]: first=[]
         second=[]
         third=[]
         fourth=[]
         for index,row in test_data.iterrows():
             first.append(split(row[0])[0])
             second.append(split(row[0])[1])
             third.append(split(row[0])[2])
             fourth.append(split(row[0])[3])
```

```
In [32]: test_data_split=pd.DataFrame()
         test_data_split["First"]=first
         test_data_split["Second"]=second
         test_data_split["Third"]=third
         test_data_split["Fourth"]=fourth
```

```
In [33]: enc_test_data=enc.transform(test_data_split).toarray()
```

```
In [34]: y_test=clf.predict(enc_test_data)
```

```
In [36]: np.savetxt("test_answer.csv", y_test, delimiter=",")
```