Peerawat Pipattanakulchai
57010922

“Classification สำหรับจัดกลุ่ม Type1 ของโปเกมอน”

Introduction to Data Analytics
2nd semester of 2017

25/March/2018

1. Attribute Choosing and Data Cleaning.

Data Cleaning

The first data problem is missing value, some rows in some columns has no value. The following code is data cleaning by replace null value with some value. In “type2” column, I fill in the ”NULL” string to the missing value. In the “abilities” column, I slice string to one first pokemon ability in each row.

The second problem is type data is string, It can’t use in data prediction (“type1”, “type2”, “abilities”). I have to encode them to number

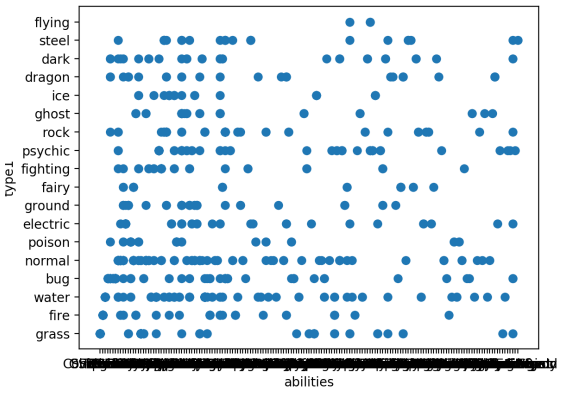
```
usedAttributes = ["type1", "abilities",'against_dark', 'against_bug', 'base_total',
                  'base_egg_steps','against_dragon','against_electric','against_fairy',
                    'against_fight','against_fire','against_flying','against_ghost','against_grass',
                    'against_ground','against_ice','against_psychic','against_rock','against_steel',
                    'against_water','type2']  # Define read attributes.
pokemonData = pd.read_csv('pokemon.csv', usecols=usedAttributes)  # Read CSV file.

    # Replace NaN object with "NULL" string
for i in range(usedAttributes.__len__()):   # attribute iterator
    for j in range(pokemonData.__len__()):      # row iterator

        if pokemonData[usedAttributes[i]][j] is pd.np.nan:   # Replace Nan Object with "NULL"
            pokemonData[usedAttributes[i]][j] = "NULL"

        if i == 1 :                 # Clean "abilities" value to one first ability
            str_index = 0
            for charac in pokemonData[usedAttributes[i]][j]:
                if charac == '\'' and str_index > 2 :
                    pokemonData[usedAttributes[i]][j] = pokemonData[usedAttributes[i]][j][2:str_index]
                    break
                str_index = str_index + 1

########## Encoded catagory/string  to number

type1_value = dict(zip(pokemonData['type1'].astype('category').cat.categories.tolist(), range(18)))
pokemonData['type1_encoded'] = pokemonData['type1'].map(type1_value)

type2_value = dict(zip(pokemonData['type2'].astype('category').cat.categories.tolist(), range(19)))
pokemonData['type2_encoded'] = pokemonData['type2'].map(type2_value)

#print(pokemonData['abilities'].astype('category').values)
abilities_value = dict(zip(pokemonData['abilities'].astype('category').cat.categories.tolist(), range(165)))
pokemonData['abilities_encoded'] = pokemonData['abilities'].map(abilities_value)
```

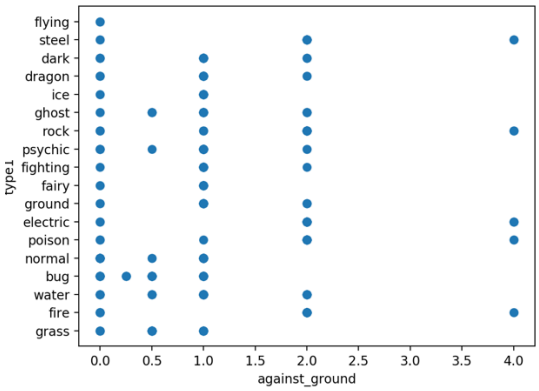
Attribute Choosing

Deciding which attribute to use for model training by shape of scatter-plot. The following code generate scatter-plot between “type1” column and another column.

```
for i in range(usedAttributes.__len__()):
    if( i != 0):
        plt.xlabel(usedAttributes[i]) # X label for scatter plot
        plt.ylabel(usedAttributes[0]) # Y label for scatter plot
        plt.scatter(pokemonData[usedAttributes[i]], pokemonData[usedAttributes[0]]) # Scatter Plot
        plt.show()
```

And this is some chart from the code.





The attribute I use is

```
X = pokemonData[['type2_encoded', 'abilities_encoded','against_dark', 'against_bug', 'base_total',
                 'base_egg_steps','against_electric','against_fairy','against_fight','against_fire',
                 'against_flying','against_ghost','against_grass','against_ground','against_ice',
                 'against_psychic','against_rock','against_steel','against_water']]
```

2. Logistic Regression

Code

```
49  ########### Logistic Regression Classification
50
51  from sklearn.linear_model import LogisticRegression
52  from sklearn import metrics
53  from sklearn import model_selection
54  from sklearn.model_selection import train_test_split
55
56  X = pokemonData[['type2_encoded', 'abilities_encoded','against_dark', 'against_bug', 'base_total',
57                   'base_egg_steps','against_electric','against_fairy','against_fight','against_fire',
58                   'against_flying','against_ghost','against_grass','against_ground','against_ice',
59                   'against_psychic','against_rock','against_steel','against_water']]
60  # X = pokemonData[['against_ground' , 'against_water' , 'against_grass',
61  #              'against_ghost', 'against_ice' , 'against_rock' ,'against_fire']]
62  y = pokemonData['type1_encoded']
63  clf = LogisticRegression()
64  kfold = model_selection.KFold(n_splits=10, random_state=7)
65  results = model_selection.cross_val_score(clf, X, y, cv=kfold, scoring= 'accuracy')
66  print("10-fold accuracy: %.3f" % (results.mean()))
```

Accuracy

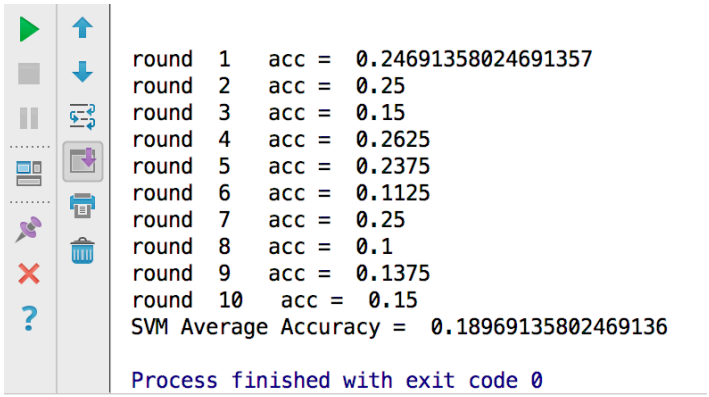
```
10-fold accuracy: 0.855
```

3. SVM

Code

```
########## SVM
from sklearn import svm
from sklearn.model_selection import KFold

kf = KFold(n_splits=10)
clf = svm.SVC()
i = 1
sum_acc = 0
for train, test in kf.split(X):
    X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.iloc[test]
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    acc = np.sum(predictions == y_test)/len(predictions)
    sum_acc += acc
    print("round ", i, "  acc = ", acc)
    i = i+1

print("Average Accuracy = ",sum_acc/10)
```

Accuracy



```
round  1   acc =  0.24691358024691357
round  2   acc =  0.25
round  3   acc =  0.15
round  4   acc =  0.2625
round  5   acc =  0.2375
round  6   acc =  0.1125
round  7   acc =  0.25
round  8   acc =  0.1
round  9   acc =  0.1375
round  10   acc =  0.15
SVM Average Accuracy =  0.18969135802469136

Process finished with exit code 0
```

4. Logistic Regression Model

Final Trained Model Code

```
88  ########## Final trained Model of Logistic Regression
89  X_train, X_test, y_train, y_test = train_test_split(X,y)
90  clf = LogisticRegression()
91  clf.fit(X_train, y_train)
92  predictions = clf.predict(X_test)
93  acc = np.sum(predictions == y_test)/len(predictions)
94  print("Logistic R. Accuracy = ", acc)
95  print("Coeficient:\n",clf.coef_)
96  print("\nIntercept:\n",clf.intercept_)
```

Model (Coefficient & Intercept)

This coefficient list is prediction model, The row is in order of pokemon type and the column is in order of

```
Coeficient:
 [[ 6.97153487e-02  5.88831992e-03 -7.94561472e-01 -7.24352866e-01
  -4.94612527e-03 -3.10839003e-05 -1.03244339e+00  2.40792119e-01
  -1.99113055e+00  1.78864668e+00  1.56112335e+00 -4.67622193e-01
  -6.30911075e-01 -6.66125793e-01 -1.67732773e+00 -4.84561886e-01
   1.23751695e+00 -7.63596575e-01  6.79482194e-01]
 [ 1.03942871e-01 -8.30801571e-03 -1.22088395e+00  7.37850386e-01
  -1.42765476e-03  2.56400374e-06  8.80612827e-02  1.39923404e+00
   4.63757476e-01 -4.10222420e-01 -8.06733749e-01 -5.38284890e-01
  -1.02989783e+00 -5.18326756e-01 -6.21558657e-01 -2.37706348e+00
   3.87581912e-01 -1.79253920e-01  1.82070467e-01]
 [ 3.73322921e-02  1.68349822e-02 -1.51474248e-01 -7.46965160e-01
  -1.69535050e-04  1.01054568e-04 -1.61013494e+00  1.06268016e+00
  -4.91437079e-01 -1.89386147e+00 -4.78862303e-01  7.47404145e-02
  -1.60957477e+00 -1.40112794e-01  4.84656187e-01  3.62766297e-01
  -3.72212775e-01  1.05987336e-01 -1.96800959e+00]
 [-1.10198501e-01  9.36443744e-03 -1.53398332e-01  1.87552500e-01
  -1.05170747e-03  1.39053925e-05 -1.77388910e+00  1.87283276e-01
  -5.97250252e-01  6.82932094e-01 -2.94400095e+00 -2.36356442e-01
  -5.53669142e-02  1.36962175e+00  4.03615753e-01  1.86927286e-01
   8.26013161e-01 -2.15761008e+00 -9.53210965e-01]
 [-6.03379472e-01 -3.62543783e-03 -1.57969818e+00 -1.34569154e+00
   3.22722496e-03 -1.10625535e-05  5.20907786e-01 -3.46735563e-01
  -1.64743370e+00 -2.15867116e-01 -6.17560436e-01 -8.98593882e-02
  -4.36892120e-01 -9.74192529e-01  1.60337387e-01 -3.30206720e-01
  -7.27843840e-02  2.15252281e+00 -2.88023180e-01]
 [-6.99603030e-02  8.38489531e-03 -1.18624568e+00 -1.31629819e+00
  -3.06629015e-03 -1.80291466e-04 -3.77444318e-01  1.45989899e+00
   2.88015078e-01  3.95847868e-01  5.20048896e-01  1.34323033e+00
  -3.59240067e-01 -6.46533204e-01 -1.55756155e+00  6.84898033e-01
  -2.41475580e+00  1.95669590e-01  3.50786824e-01]
 [-4.43202398e-02 -6.93487637e-03 -7.16223383e-01 -1.03363479e+00
   5.93883664e-03 -2.55105704e-05 -1.88750918e-01 -9.14034207e-01
  -1.01959633e-01 -1.66540352e+00 -1.33823827e-01 -2.98418926e-01
  -9.02505927e-01  6.15229861e-02 -1.27862802e+00 -1.64017897e-01
   4.49566857e-01 -1.26454263e+00  1.69619762e+00]
 [-4.11098039e-01 -3.03848596e-02 -4.17659777e-02 -5.44639547e-01
  -1.80557433e-03 -5.06398509e-04  2.89102471e-02  5.92863082e-01
  -4.04469680e-01 -6.98360007e-01 -4.13669031e-01  6.57164618e-02
  -4.13339267e-01 -6.05917172e-01  1.43501886e+00 -7.62221180e-02
   4.73697264e-01 -8.44937002e-02 -3.83962429e-01]
 [-9.83861836e-02 -1.44750161e-02  1.81638314e+00 -1.91072697e+00
  -3.23559517e-04 -3.41370218e-04 -5.36391345e-01 -2.42268171e-01
  -2.11185339e+00 -1.46785120e-01  7.92482163e-02  1.82879006e+00
  -4.49820093e-01 -5.33704514e-01  1.53563170e-01  2.44218298e-01
  -5.33763741e-01  1.97045257e-01 -2.04820309e-01]
 [-1.05924827e-01 -4.92004218e-03 -9.13716674e-01  1.45927299e+00
   7.17156293e-04 -2.13689276e-04 -9.12301859e-01 -1.76862430e+00
   1.74231051e-01  1.13365580e+00 -5.55774379e-01  5.57739166e-01
  -1.40721335e+00 -5.97308040e-01  1.40665326e+00  1.16932210e+00
  -1.66480965e+00  1.06600715e+00 -2.11929319e+00]
```

```
 [-1.05810646e-01 -4.10179489e-03 -1.17509493e-01 -1.35981142e-01
  -2.88433715e-04 -9.91311353e-06 -1.80589905e+00 -2.75854661e-01
  -3.94891507e-01  2.87307222e-01 -6.76142718e-01  1.18474809e-01
   3.40172014e-01 -5.49182537e-01  1.11648587e+00 -6.85112329e-01
  -1.52763763e+00 -6.44063588e-01  1.11187844e+00]
 [-4.74320384e-02 -5.33235124e-03  4.02655707e-01 -4.90966170e-01
  -4.98664017e-03 -7.89967993e-06 -6.07065764e-01 -4.92039111e-01
   5.26825626e-01  1.00533978e+00 -1.49184886e+00  6.54660446e-01
   5.73162669e-01 -1.13025495e+00 -2.49922454e+00 -1.38457702e-01
   1.51496329e+00  1.45907265e+00 -1.05519710e+00]
 [-1.04187846e-01 -7.43202597e-04  5.31433903e-01 -4.55206068e-01
  -8.63053178e-04 -3.85375360e-06  4.49350181e-01 -5.20612583e-01
   1.59310788e+00 -2.68151315e-01 -3.67724408e-01 -3.34167622e+00
  -4.73756920e-01 -1.11379755e+00  4.80041954e-01  4.39885743e-01
  -1.24913574e-01  2.17246124e-01 -3.14882705e-02]
 [-1.26771721e-01  6.25184253e-03 -7.71786269e-01 -5.79920276e-01
  -1.25109246e-03 -1.41806003e-04  7.48640696e-01 -1.20818265e+00
  -1.49408093e+00 -4.00785398e-01 -4.30615098e-01 -8.14877022e-01
  -1.67211771e+00  1.75022604e+00  4.55097669e-01  1.36600901e+00
  -2.60660190e-01  4.62538909e-01 -9.38178782e-01]
 [-2.43924463e-01  1.79891056e-02  8.79558934e-01  1.57217347e+00
  -1.42589587e-03  8.33820235e-05  1.29850235e-01 -1.22271425e+00
  -1.14202041e+00 -7.59294734e-01 -1.40577056e+00  1.90028870e+00
  -1.31632308e+00 -1.14949057e+00 -5.77991227e-01 -2.48113957e+00
  -3.72474830e-01  7.23342345e-01  1.60679241e-01]
 [-1.55879609e-02 -1.29555326e-03 -6.15950606e-01 -1.87214825e-01
  -3.70192810e-03  5.39168354e-05  3.00342072e-01 -5.27408252e-01
   8.58513876e-01 -2.39932195e+00 -2.30415033e+00  2.63287007e-01
   2.02348007e-01 -1.39719570e-01 -1.86709400e-01 -1.80081177e-01
  -3.94412583e-02  2.54453350e+00  1.19281883e-01]
 [-1.40885727e-01 -3.59547447e-03  3.22021556e-01 -1.35100169e+00
  -1.90792851e-03  5.96843919e-05  1.11186294e+00 -1.15619751e+00
   6.21025893e-01  9.11975885e-01 -1.40871797e+00  1.02448929e+00
  -1.08934484e+00  3.25740472e-01 -7.59898415e-01 -1.09080501e+00
  -1.83114258e+00 -1.69933877e-01  2.97947460e-01]
 [-1.14616516e-01  3.98463682e-03 -3.76864642e-01  3.76640632e-01
   6.19639982e-03 -1.27769518e-04  1.63158992e+00 -4.38471551e-01
  -8.54679208e-01 -1.19315270e+00  4.77851925e-01  1.24367482e-01
   1.68971003e+00 -9.24835189e-01 -1.47165398e+00  1.91561063e-01
  -4.29943997e-01 -2.16324642e+00 -2.35252293e+00]]

Intercept:
 [-0.26957103 -0.14473642 -0.3545808   0.23060902  0.03057975 -0.22913012
 -0.27219603 -0.0607512  -0.11443158  0.01363328 -0.10533565 -0.49462275
  0.11840677 -0.00524146 -0.56212778 -0.9445556   0.01658811  0.08994404]

Process finished with exit code 0
```

5. Accuracy of each class

Logistic Regression accuracy

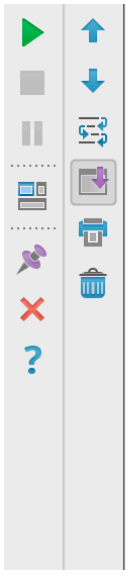
```
105 from sklearn.metrics import classification_report
106 print(classification_report(y_test, predictions,target_names=list(type1_value.keys())))
```

```
             precision    recall  f1-score   support

        bug       0.81      1.00      0.90        13
       dark       1.00      0.62      0.77         8
     dragon       0.70      0.78      0.74         9
   electric       0.75      0.90      0.82        10
      fairy       1.00      1.00      1.00         3
   fighting       0.73      1.00      0.84         8
       fire       1.00      1.00      1.00        13
     flying       0.83      0.71      0.77         7
      ghost       1.00      0.94      0.97        17
      grass       0.80      0.73      0.76        11
     ground       1.00      0.71      0.83         7
        ice       1.00      1.00      1.00        30
     normal       0.50      0.67      0.57         6
     poison       1.00      1.00      1.00        17
    psychic       0.80      1.00      0.89         8
       rock       0.89      0.80      0.84        10
      steel       0.95      0.79      0.86        24

avg / total       0.90      0.89      0.89       201
```

6. Confusion Metrix

Logistic Regression Confusion Metrix

```
101 from sklearn.metrics import confusion_matrix
102 confusion_matrix = confusion_matrix(y_test, predictions)
103 print(confusion_matrix)
```



```
[[20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  7  0  0  0  1  0  0  0  1  0  0  0  0  1  0  0]
 [ 0  0  1  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  0  0  0  6  0  0  0  0  0  1  0  0  0  0  1  0]
 [ 0  0  0  1  0  0 14  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 18  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 1  0  0  1  0  0  0  0  0  0  0  0  0  5  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 16  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 11  2  2]
 [ 0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  2  5  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15]]
```