Report On Models and Techniques used in Kaggle Competition

Murali Krishna
MT19132

- I have used three machine learning models :
  - CNN
  - Random Forest
  - XGBoost .

- The above models are applied by trying various parameters tuning which are specified below.

- Out of all the models applied, *XGBoost* with certain parameters (given below) yielded a high f1 score for both validation data and kaggle scoreboard. The kaggle public score for this model gave a 51.514 percentage score.
- I have tried by tuning various parameters for all the three models that I have tried. But it has never crossed 55 percent f1 score on validation data.

Preprocessing Methods Used

- The data contain few columns with complete zero's . Out of 46 columns (without labels), there were 6 columns that contained complete zero's.
  So For few models I have reduced the features to 40 and for some model have taken all 46 features.

- Have used **two feature reduction techniques** i.e *PCA* and *SelectKBest* . Have tried by reducing the features to 10, 15 , 25 etc.

- Have used **two Standardization/Normalization** mechanisms : MinMaxScaler and StandardScaler to adjust ranges of the data.

Explanation of Models Used

- **CNN :** Have used sequential with 3 dense layers with input shape (Number of attributes) with activation layers as relu, relu and softmax. Optimizer used is ADAM and the metric is 'accuracy'.

```
model = Sequential([
                    Dense(16, input_shape = (46,), activation='relu'),
                    Dense(32, activation = 'relu'),
                    Dense(3, activation = 'softmax')
                    ])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 16)                752
_________________________________________________________________
dense_4 (Dense)              (None, 32)                544
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 99
=================================================================
Total params: 1,395
Trainable params: 1,395
Non-trainable params: 0
_________________________________________________________________
```

Have used all the different preprocessing techniques mentioned above.
Using CNN i have received low scores i.e 41 percent in kaggle public scoreboard.
So as the scores received are very less, I tried out other models.

- **Random Forest** : Random Forest performed better than CNN. I have tried random forest by changing various parameters such as n_estimators, max_depth, class_weight, n_jobs, max_leaf nodes.
  As the data is imbalanced i.e classes with 0 are more than compare others. So have used a parameter “class_weight = balanced”.
  Have tried Random Forest with all feature reduction and normalization techniques.

```
1 from sklearn.ensemble import RandomForestClassifier
```

```
1 rand_forest = RandomForestRegressor(n_estimators=90,random_state=42,max_depth=15)
```

```
1 rand_forest.fit(x_train_rf, y_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=15, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=90, n_jobs=None, oob_score=False,
                      random_state=42, verbose=0, warm_start=False)
```

By applying random forest, the score(validation) yielded was in the range 45 to 53 percent.
The best results were yielded from XGBoost Classifier.

- **XGBoost** : As we are trying to learning about the rankings of a particular dataset, XGboost is suitable for this type of problems . In wikipedia page of “Learning to rank”, I have found that XGBoost is a type of Ranking technique. And this model yielded highest score than the above two models.
  I have tried various parameters and finally found the below parameters as more effective in terms of scores
  Parameters listed has yielded highest score used are :

```
[ ]  1 param = {'n_estimators':1500,'max_depth': 60,   'objective': 'multi:softprob',  'num_class': 3,
     2         'eval_metric':'auc','scale_pos_weight':5,'eta': 0.9,  'silent': 1}
     3
```

```
[ ]  1 cls = xgb.train(param, x_train, num_round = 1000)
```

```
[ ]  1 predictions = cls.predict(x_test)
     2 xgb_pred = np.asarray([np.argmax(line) for line in preds])
```

```
1 print(accuracy_score(y_val , xgb_pred))
2 print(f1_score(y_val, xgb_pred, average='macro'))
```

```
0.785406492387245
0.5405047213775257
```

References :

For XGBoost : https://www.datacamp.com/community/tutorials/xgboost-in-python