Otto Group Product Classification Challenge

In this competition we have 93 anonimized features describing products of Otto Group. The challenge is to classify products into correct categories. Features represent counts of different events. There is little which can be done with features themselves - no high correlation between features, no skeweresness. So this competition is about modelling.

The metric to calculate the accuracy of predictions is multiclass loss. To simplify I'll calculate log loss.

As this is multiple classufication problem, the approach differs from binary classification. The result of prediction is probabilities of belonging to each class for products.

In [1]:
```
import pandas as pd
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import log_loss
from sklearn.calibration import CalibratedClassifierCV
import numpy as np
import xgboost as xgb
```

In [2]:
```
data = pd.read_csv('../input/train.csv')
test = pd.read_csv('../input/test.csv')
```

In [3]:
```
X_train = data.drop('id', axis=1)
X_train = X_train.drop('target', axis=1)
Y_train = LabelEncoder().fit_transform(data.target.values)
X_test = test.drop('id', axis=1)
```

In [4]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(X_train, Y_train, test_size=
```

At first I tried several other models (logistic regression, SVM and others), but they weren't good enough. Random Forest proved to be better. Parameters were obtained with CVgridsearch

In [6]:
```
clf = RandomForestClassifier(n_estimators=300, n_jobs=-1, criterion = 'gini'
calibrated_clf = CalibratedClassifierCV(clf, method='isotonic', cv=5)
calibrated_clf.fit(Xtrain, ytrain)

y_val = calibrated_clf.predict_proba(Xtest)
y_submit = calibrated_clf.predict_proba(X_test)
print("Loss on validation set: ", log_loss(ytest, y_val, eps=1e-15, normaliz
```

```
Loss on validation set:  0.488063577439
```

I decided to add XGBoost to improve the model and it helped. The final result was using weighted predictions from both models.

```
In [7]: params = {"objective": "multi:softprob", "num_class": 9}
        gbm = xgb.train(params, xgb.DMatrix(X_train, Y_train), 20)
        Y_pred = gbm.predict(xgb.DMatrix(X_test))
```

```
In [8]: y = 0.2 * Y_pred + 0.8 * y_submit
```

```
In [9]: sample = pd.read_csv('../input/sampleSubmission.csv')
        preds = pd.DataFrame(y, index=sample.id.values, columns=sample.columns[1:])
        preds.to_csv('Otto.csv', index_label='id')
```

This competition has already ended, but people still can submit their solutions and see their scores. Top places have score of ~0.38.

My ensemble model got a score of 0.48926.