In [7]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import os

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow import keras
from sklearn.decomposition import PCA
from sklearn.ensemble import StackingClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import log_loss
```

In [8]:
```python
os.getcwd()
```

Out[8]: 'C:\\Users\\robtu'

In [9]:
```python
os.chdir(r'C:\Users\robtu\Kaggle Competitions')
```

In [11]:
```python
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

test_ids = test['id']
test = test.drop(['id'],axis=1)
```

In [12]:
```python
train.shape
```

Out[12]: (100000, 52)

In [5]:
```python
le = LabelEncoder()
train.target = le.fit_transform(train.target)
features = train.columns[1:51]
train[features].head()
```

Out[5]:

	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	feature_9	...	feature_40	feature_41
0	0	0	1	0	1	0	0	0	0	0	...	3	0
1	0	0	0	0	2	1	0	0	0	0	...	0	0
2	0	0	0	0	0	0	0	0	0	2	...	0	0
3	0	0	0	0	0	0	0	3	0	0	...	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 50 columns

In [6]:
```python
#train.head()
```

In [14]:
```python
y_train = train['target']
X_train = train.drop(['target','id'], axis=1)
df = X_train.append(test)
```

In [5]:
```python
#df.info()
```

PCA Dimension Reduction

In [178]:
```python
pca = PCA(n_components = 45)

pca_trans = pca.fit_transform(df)
```

In [179]:
```python
pca_df = pd.DataFrame(data=pca_trans)
```

In [180]:
```python
X_train = pca_df[0:100000]
test = pca_df[100000:150000]
```

In [181]:
```python
X_train.head()
```

Out[181]:

	0	1	2	3	4	5	6	7	8	9	...	35	36
0	-3.005962	-1.635621	-2.561625	-0.639469	-0.885238	-1.176251	-0.682512	0.611503	-1.025236	-0.197477	...	-0.525330	-0.539287
1	-2.982253	-1.750327	-2.564989	-0.725127	-1.032808	-1.269029	-0.878316	1.152172	-1.561140	-1.233076	...	-0.374002	-0.448865
2	0.072336	-0.493942	-1.541736	-0.534115	-0.977030	-1.100344	-0.653076	-0.237646	-0.663316	1.529694	...	0.518562	0.543679
3	0.892799	-0.225742	4.462836	-0.651285	-1.032219	-1.348710	-0.881347	-0.435393	1.744973	-0.353929	...	-0.392054	-0.439311
4	-2.033965	-1.593626	-0.528681	-0.731055	-0.988482	-0.364498	-1.022103	-0.809601	-1.278326	-0.207429	...	-0.737722	-0.597776

5 rows × 45 columns

In [53]:
```python
X_train = df[0:100000]
test = df[100000:150000]
```

In [15]:
```python
X, X_test, y, y_test = train_test_split(X_train, y_train, test_size=0.33, random_state=42)
```

TensorFlow Model

In [60]:
```python
import tensorflow as tf
from tensorflow import keras
from sklearn.preprocessing import LabelEncoder
```

In [61]:
```python
encoder = LabelEncoder()
encoder.fit(y_train)
encoded_Y = encoder.transform(y_train)

dummy_y = tf.keras.utils.to_categorical(encoded_Y)
```

In [62]:
```python
X_train.shape
```

Out[62]: (100000, 40)

In [63]:
```python
dummy_y.shape
```

Out[63]: (100000, 4)

In [65]:
```python
X_tens = X_train.to_numpy()
test_tens = test.to_numpy()
```

In [66]:
```python
X_conv = X_tens.reshape(X_tens.shape[0],X_tens.shape[1],1)
test_conv = test_tens.reshape(test.shape[0],test.shape[1],1)
```

In [67]:
```python
X_conv.shape
```

Out[67]: (100000, 40, 1)

In [68]:
```python
test_conv.shape
```

Out[68]: (50000, 40, 1)

In [81]:
```python
model = keras.Sequential([

    keras.layers.Dense(25, activation='sigmoid', input_shape= (42,)),
    keras.layers.Dense(15, activation='sigmoid'),
    keras.layers.Dense(4, activation='sigmoid'),

   ])

model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_44 (Dense)             (None, 25)                1075      
_________________________________________________________________
dense_45 (Dense)             (None, 15)                390       
_________________________________________________________________
dense_46 (Dense)             (None, 4)                 64        
=================================================================
Total params: 1,529
Trainable params: 1,529
Non-trainable params: 0
_________________________________________________________________
```

In [4]:
```python
#model2 = keras.Sequential([

    #keras.layers.Conv1D(filters=10, kernel_size=3,padding='same', activation='relu',input_shape=(40,
1)),
    #keras.layers.Dense(40, activation='relu'),
   # keras.layers.MaxPooling1D(),
   # keras.layers.Flatten(),
    #keras.layers.Dense(35,activation='relu'),
   # keras.layers.Dense(20,activation='relu'),
   # keras.layers.Dense(10,activation='relu'),
   # keras.layers.Dense(4,activation='softmax')

#])



#metrics = [tf.keras.metrics.CategoricalCrossentropy()]
#loss = tf.keras.losses.CategoricalCrossentropy(from_logits=False,label_smoothing=0,reduction="auto",na
me="categorical_crossentropy")

#model2.compile(optimizer='Adamax',loss=loss,metrics=metrics)

#model2.summary()
```

In [1]:
```python
#history = model.fit(X_tens, dummy_y, validation_split = 0.3, epochs=3, shuffle=True)
```

In [2]:
```python
 #history2 = model2.fit(X_conv, dummy_y, validation_split=0.3, epochs=15, shuffle=True)
```

In [354]:
```python
y_pred = model.predict(test)
```

In [104]:
```python
y_pred2 = model2.predict(test_conv)
```

In [105]:
```python
y_pred2.shape
```

Out[105]: (50000, 4)

XGBoost Catboost and LightGBM Models

In [16]:
```python
import xgboost as xgb
from xgboost import plot_importance, XGBClassifier
from catboost import CatBoostClassifier, Pool
import lightgbm as lgb
from lightgbm import LGBMClassifier
```

In [17]:
```python
xgb_params = {
    'n_estimators':1000,
    'learning_rate':0.746463,
    'max_depth':1,
    'lambda':25.46112,
    'random_state':21,
    'objective':'multi:softprob',
    'eval_metric':'mlogloss',
}
```

In [18]:
```python
clf_xgb = xgb.XGBClassifier(**xgb_params)

clf_xgb.fit(X,
            y,
            verbose=False,
            ## the next three arguments set up early stopping.
            early_stopping_rounds=30,
            eval_metric=['mlogloss'],
            eval_set=[(X, y),(X_test, y_test)]
           )
```

Out[18]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eval_metric='mlogloss',
              gamma=0, gpu_id=-1, importance_type='gain',
              interaction_constraints='', lambda=25.46112,
              learning_rate=0.746463, max_delta_step=0, max_depth=1,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=1000, n_jobs=0, num_parallel_tree=1,
              objective='multi:softprob', random_state=21, reg_alpha=0,
              reg_lambda=25.4611206, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [17]:
```python
X.shape
```

Out[17]: (67000, 50)

In [11]:
```python
train_pool = Pool(data=X, label=y)
test_pool = Pool(data=X_test, label=y_test.values)
```

In [12]:
```python
params_cb = {
    'n_estimators' : 9000,
    'od_wait' : 300,
    'loss_function': 'MultiClass',
    'eval_metric': 'MultiClass',
    'learning_rate' : 0.0165847,
    'reg_lambda': 17.7924786,
    'subsample': 0.537623 ,
    'depth': 2,
    'min_data_in_leaf': 19,
    'verbose':False,
    'bootstrap_type': 'Bernoulli',
    'random_state' :42,
    'leaf_estimation_method':'Newton',
}

params_cb2 = {
    'iterations': 17000,
    'learning_rate': 0.01,
    'depth': 4,
    'loss_function':'MultiClass',
    'od_wait': 1000,
    'od_type' : 'Iter',
    'min_data_in_leaf' : 1,
    'max_ctr_complexity': 15,

}
```

In [13]:
```python
catmod = CatBoostClassifier(**params_cb)


catmod.fit(train_pool,verbose=1100,plot=True,eval_set=test_pool)
```

```
0:      learn: 1.3764564        test: 1.3763699 best: 1.3763699 (0)     total: 154ms    remaining: 23
m 4s
1100:   learn: 1.0999140        test: 1.0953577 best: 1.0953577 (1100)  total: 42.5s    remaining: 5m
4s
2200:   learn: 1.0931169        test: 1.0906799 best: 1.0906799 (2200)  total: 1m 26s   remaining: 4m
27s
3300:   learn: 1.0894625        test: 1.0889359 best: 1.0889359 (3300)  total: 2m 14s   remaining: 3m
52s
4400:   learn: 1.0869065        test: 1.0880714 best: 1.0880697 (4396)  total: 3m 5s    remaining: 3m
14s
5500:   learn: 1.0848732        test: 1.0875711 best: 1.0875711 (5500)  total: 4m 1s    remaining: 2m
33s
6600:   learn: 1.0831345        test: 1.0872284 best: 1.0872284 (6600)  total: 5m 6s    remaining: 1m
51s
7700:   learn: 1.0816585        test: 1.0870227 best: 1.0870219 (7698)  total: 6m 6s    remaining: 1m
1s
8800:   learn: 1.0802442        test: 1.0869398 best: 1.0869363 (8779)  total: 7m 10s   remaining: 9.
74s
8999:   learn: 1.0800280        test: 1.0869222 best: 1.0869194 (8963)  total: 7m 22s   remaining: 0u
s

bestTest = 1.086919352
bestIteration = 8963

Shrink model to first 8964 iterations.
```

Out[13]: <catboost.core.CatBoostClassifier at 0x25b486d6370>

In [14]:
```python
cat_pred1 = catmod.predict_proba(X_test)
cat_pred2 = np.clip(cat_pred1, 0.08, 0.95)
log_loss(y_test,cat_pred2)
```

Out[14]: 1.0866426451546274

In [22]:
```python
params_lgbm = {
    'learning_rate': 0.03602375,
    'max_depth': 2,
    'min_child_samples':61,
    'min_child_weight' : 0.2569581,
    'metric': 'multi_logloss',
    'random_state': 42,
    'n_estimators': 10000,
    'objective': 'multiclass',
}
```

In [19]:
```python
#lgbmod = lgb.LGBMClassifier(**params_lgbm)


#lgbmod.fit(
            #X,
            #y,
            #eval_set=[(X_test, y_test)],
            #early_stopping_rounds=500,
            #verbose=True)
```

In [18]:
```python
cat_predictions = catmod.predict_proba(test)
cat_predictions2 = np.clip(cat_predictions, 0.08, 0.95)
```

In [172]:
```python
predictions = clf_xgb.predict_proba(X_test)
log_loss(y_test,predictions)
```

Out[172]: 1.1031687413697893

Stacking Classifier

In [24]:
```python
estimators = [('lgbm',LGBMClassifier(**params_lgbm)),('cb',CatBoostClassifier(**params_cb)),('xgb',XGBC
lassifier(**xgb_params))]

stack_mod = StackingClassifier(

        estimators=estimators,
        final_estimator=LGBMClassifier(),
        stack_method='predict_proba',
        n_jobs=-1
)
```

In [3]:
```python
#stack_mod.fit(X,y)

#val = stack_mod.predict_proba(X_test)
#log_loss(y_test,val)
```

In [38]:
```python
stack_pred = stack_mod.predict_proba(test)
```

In [19]:
```python
df_results2 = pd.DataFrame({'id':test_ids, 'Class_1':cat_predictions2.T[0], 'Class_2':cat_predictions2.
T[1], 'Class_3':cat_predictions2.T[2] ,'Class_4':cat_predictions2.T[3]})
df_results2.head()
```

Out[19]:

	id	Class_1	Class_2	Class_3	Class_4
0	100000	0.092863	0.638704	0.144634	0.123800
1	100001	0.080000	0.706920	0.130900	0.084291
2	100002	0.092345	0.630468	0.183383	0.093804
3	100003	0.097111	0.536670	0.276161	0.090058
4	100004	0.080000	0.604472	0.186167	0.130002

In [21]:
```python
compression_opts = dict(method='zip',
                        archive_name='results24.csv')

df_results2.to_csv('results24.zip', index=False,
          compression=compression_opts)
```