KaggleCompetition

December 27, 2018

1 Kaggle Competition

1.1 Rebekah Griesenauer

1.1.1 Run useful functions

In [1]:
```
from sklearn import preprocessing
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import shutil
import os
import requests
import base64

# TensorFlow with Dropout for Regression
##############################################
%matplotlib inline
from matplotlib.pyplot import figure, show
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn import metrics
from scipy.stats import zscore
from keras.callbacks import EarlyStopping
from keras.layers import Dense, Dropout
from keras import regularizers
from keras.models import Sequential


# Encode text values to dummy variables(i.e. [1,0,0],[0,1,0],[0,0,1] for red,green,blue)
def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = "{}-{}".format(name, x)
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)
```

```
        # Encode a numeric column as zscores
        def encode_numeric_zscore(df, name, mean=None, sd=None):
            if mean is None:
                mean = df[name].mean()

            if sd is None:
                sd = df[name].std()

            df[name] = (df[name] - mean) / sd


        # Convert all missing values in the specified column to the median
        def missing_median(df, name):
            med = df[name].median()
            df[name] = df[name].fillna(med)

        # Convert a Pandas dataframe to the x,y inputs that TensorFlow needs
        def to_xy(df, target):
            result = []
            for x in df.columns:
                if x != target:
                    result.append(x)
            # find out the type of the target column.  Is it really this hard? :(
            target_type = df[target].dtypes
            target_type = target_type[0] if hasattr(target_type, '__iter__') else target_type
            # Encode to int for classification, float otherwise. TensorFlow likes 32 bits.
            if target_type in (np.int64, np.int32):
                # Classification
                dummies = pd.get_dummies(df[target])
                return df.as_matrix(result).astype(np.float32), dummies.as_matrix().astype(np.fl
            else:
                # Regression
                return df.as_matrix(result).astype(np.float32), df.as_matrix([target]).astype(np
```

```
Using TensorFlow backend.
```

In [2]:
```
        # Setup path and read in data
        path = "./data/all/"

        filename_read_train = os.path.join(path,"train.csv")
        filename_read_test = os.path.join(path,"test.csv")
        df_train = pd.read_csv(filename_read_train,na_values=['NA', '?'])
        df_test = pd.read_csv(filename_read_test,na_values=['NA', '?'])

        ids = df_test['id']
        df_test.drop('id',1,inplace=True)
```

```
df_train.drop('id',1,inplace=True)

df_train = df_train.reindex(np.random.permutation(df_train.index))
df_train.reset_index(inplace=True, drop=True)

density_gold = 19.32
density_platinum = 21.09
density_bronze = 9.29
density_tin =  7.31
density_silver = 10.49
```

2 Feature Engineering

2.0.1 Operations performed on each feature

1. ID: Drop
2. shape: encode as text dummy variable
3. metal: encode as text dummy variable
4. metal_cost: used to calculate cost of metal/ total cost, then drop
5. height: use to calculate volume, then drop
6. width: use to calculate volume, then drop
7. length: use to calculate volume, then drop
8. led: use to calculate volume of LED, then drop
9. gears: use to calculate volume of gears, then drop
10. motors:use to calculate volume of motors, then drop
11. led_vol: calculate volume of LED (df['led']*0.027), encode as z-score
12. motor_vol: calculate volume of LED ((222) * df['motors']), encode as z-score
13. gear_vol: calculate volume of LED ((122) * df['gears']), encode as z-score
14. volume_parts: led_vol+motor_vol+gear_vol, encode as z-score
15. cost: fill missing values with median, encode as z-score
16. weight (target)

Additional Columns added: 1. volume (volume of widget): use height, width, and length along with individual equations for sphere, box, and cylinder, encode as z-score 2. est_weight (estimated weight): density of metal * volume, encode as z-score 3. price_per_metal = df['cost']/df['metal_cost'], encode as z-score 4. final_volume = df['volume']-df['volume_parts'], encode as z-score

```
In [3]: def feature_engineering_calculations(df):

            df['volume']=0

            df['volume'] = df.apply(
                lambda row: row['length']*row['height']*row['width'] if row['shape']=='box' else

            df['volume'] = df.apply(
                lambda row: row['height']*np.pi*(row['width']/2)**2 if row['shape']=='cylinder'
```

```
    df['volume'] = df.apply(
        lambda row: (4/3)*np.pi*(row['length']/2)**3 if row['shape']=='sphere' else row[

    df['est_weight']=0

    df['est_weight'] = df.apply(
        lambda row: density_gold*row['volume'] if row['metal']=='gold' else row['est_wei

    df['est_weight'] = df.apply(
        lambda row: density_platinum*row['volume'] if row['metal']=='platinum' else row[

    df['est_weight'] = df.apply(
        lambda row: density_bronze*row['volume'] if row['metal']=='bronze' else row['est

    df['est_weight'] = df.apply(
        lambda row: density_tin*row['volume'] if row['metal']=='tin' else row['est_weigh

    df['est_weight'] = df.apply(
        lambda row: density_silver*row['volume'] if row['metal']=='silver' else row['est

    df['led_vol'] = df['led']*0.027 # 0.27 by dividing not null values in led by the led

    df['motor_vol'] = (2*2*2) * df['motors']

    df['gear_vol'] = (1*2*2) * df['gears']

    missing_median(df,'cost')

    df['price_per_metal'] = df['cost']/df['metal_cost']

    df['volume_parts'] = df['led_vol'] + df['motor_vol'] + df['gear_vol']

    df['final_volume'] = df['volume']- df['volume_parts']
    df['final_volume'] = df.apply(lambda row: 0 if row['final_volume']<0 else row['final

    return df
```

In [4]:
```
def feature_engineering_encode(df):
    encode_text_dummy(df,"shape")
    encode_text_dummy(df,"metal")
    df.drop('metal_cost',1,inplace=True)
   # df.drop('height',1,inplace=True)
   # df.drop('width',1,inplace=True)
    #df.drop('length',1,inplace=True)
    #df.drop('led',1,inplace=True)
    #df.drop('gears',1,inplace=True)
    #df.drop('motors',1,inplace=True)
```

```
            encode_numeric_zscore(df,'led_vol')
            encode_numeric_zscore(df,'motor_vol')
            encode_numeric_zscore(df,'gear_vol')
            encode_numeric_zscore(df,'volume_parts')
            encode_numeric_zscore(df,'cost')
            encode_numeric_zscore(df,'volume')
            encode_numeric_zscore(df,'est_weight')
            encode_numeric_zscore(df,'price_per_metal')
            encode_numeric_zscore(df,'final_volume')
            return df
```

In [5]:
```
df_train = feature_engineering_calculations(df_train)
df_train = feature_engineering_encode(df_train)
```

In [6]:
```
df_train['weight']=df_train['weight'].astype('float')
x,y = to_xy(df_train,"weight")
# Split into train/test
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.20, random_state=45)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:65: FutureWarning: Method .as_matri
```

In [7]:
```
model = Sequential()
model.add(Dense(200, input_dim=x.shape[1]))
model.add(Dropout(0.01))
model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(25, activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=10, verbose=1, mode
model.fit(x_train,y_train,validation_data=(x_test,y_test),callbacks=[monitor],verbose=0,
pred = model.predict(x_test)
# Measure RMSE error.
score = np.sqrt(metrics.mean_squared_error(pred,y_test))
print("Final score (RMSE): {}".format(score))
```

```
Epoch 00017: early stopping
Final score (RMSE): 207.98391723632812
```

In [8]:
```
df_test = feature_engineering_calculations(df_test)
df_test = feature_engineering_encode(df_test)
x = df_test.as_matrix().astype(np.float32)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning: Method .as_matrix
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [9]: pred = model.predict(x)
```

```
In [10]: submit_df=pd.DataFrame(pred)
         submit_df.insert(0,'id',ids)
         submit_df.columns = ['id','weight']
         submit_df=submit_df.set_index('id')
         submit_df.to_csv('kaggle_submit_df3_2.csv')
```

```
In [11]: from sklearn import metrics
         import scipy as sp
         import numpy as np
         import math
         from sklearn import metrics

         def perturbation_rank(model, x, y, names):
             errors = []

             for i in range(x.shape[1]):
                 hold = np.array(x[:, i])
                 np.random.shuffle(x[:, i])
                 #print(i)
                 pred = model.predict(x)
                 error = metrics.mean_squared_error(y, pred)

                 errors.append(error)
                 x[:, i] = hold

             max_error = np.max(errors)
             importance = [e/max_error for e in errors]

             data = {'name':names,'error':errors,'importance':importance}
             result = pd.DataFrame(data, columns = ['name','error','importance'])
             result.sort_values(by=['importance'], ascending=[0], inplace=True)
             result.reset_index(inplace=True, drop=True)
             return result
```

```
In [12]: from IPython.display import display, HTML

         names = list(df_train.columns) # x+y column names
         names.remove("weight")
         rank = perturbation_rank(model, x_test, y_test, names)
         display(rank)
```

	name	error	importance
0	volume	1.439384e+06	1.000000
1	est_weight	1.355591e+06	0.941785
2	price_per_metal	1.307640e+06	0.908472
3	gears	5.732306e+05	0.398247
4	motors	5.293460e+05	0.367759

```
5              cost  4.575491e+05    0.317878
6     volume_parts  3.304507e+05    0.229578
7          led_vol  3.224614e+05    0.224027
8              led  2.750701e+05    0.191103
9         gear_vol  2.384855e+05    0.165686
10       motor_vol  1.859649e+05    0.129198
11  metal-platinum  8.234728e+04    0.057210
12    metal-silver  8.225710e+04    0.057147
13    shape-sphere  5.626880e+04    0.039092
14       metal-tin  5.622491e+04    0.039062
15      metal-gold  5.582493e+04    0.038784
16           width  5.527263e+04    0.038400
17          length  5.509747e+04    0.038279
18    metal-bronze  5.483995e+04    0.038100
19  shape-cylinder  5.359175e+04    0.037232
20       shape-box  5.242900e+04    0.036425
21          height  4.925712e+04    0.034221
22    final_volume  4.377762e+04    0.030414
```

In [ ]: