AI4FutureWorkForce Data Ingress

This notebook takes in lightly processed data and generates dataframes for use in Data Processing.

Dependencies

```
import pandas as pd
import numpy as np
import _pickle as pickle
import itertools

from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from collections import defaultdict
from numpy.random import seed
seed(42)
```

Load Data from csv

We are placing data in the root directory however we could also use an S3 bucket or a SageMaker instance.

```
# DATA_PATH = 's3://staging-individual-786432523-eu-west-1-test1/AI4FWF/'
DATA_PATH = ''

# SAVE_PATH = '/home/ec2-user/SageMaker/s3/staging-individual-786432523-eu-west-1-test1/AI
SAVE_PATH = ''
```

```
# df_raw = pd.read_csv(DATA_PATH + "20181108_IBD_deal_data.csv", delimiter=',', encoding='
df_raw = pd.read_csv(DATA_PATH + "hack_data.csv", delimiter=',', encoding='utf-8')

# df_raw.sort_values(by='startDate',inplace=True)
```

```
df_raw.head()
```

	Fake Applicant ID	Age (Birthday Masked)	Income	Education	MAX(Learner Test Score)	Primary Interest In Course	Hours Coded	How Many Hours A Week Can You Commit To Class	E
0	58123	54	11	0	65	0	1	3	
1	35033	30	3	1	60	0	3	3	

Prep Base Data

Split into a seperate dataframe for data and labels

```
# List all columns of data table
column_list = ['Age (Birthday Masked)','Income','Education',
               'MAX(Learner Test Score)','Primary Interest In Course',
               'Hours Coded','How Many Hours A Week Can You Commit To Class',
               'Promise Zone Indicator','Hacker Rank Score', 'Location; number']

# List desired columns for train/test/validation
desired_columns = ['Age (Birthday Masked)', 'Income', 'MAX(Learner Test Score)',
                   'Hours Coded',
                   'How Many Hours A Week Can You Commit To Class',
                   'Promise Zone Indicator','Location; number']

df_data = df_raw[column_list]
df_labels = df_raw[['Completed?']]

df_data, df_validata, df_labels, df_valilabels = train_test_split(
    df_data, df_labels, test_size=0.2, random_state=42, shuffle=False)
```

```
df_labels.head()
```

	Completed?
0	0
1	0
2	0
3	0
4	0

Save Base Data

```
df_data.head(5)
```

	Age (Birthday Masked)	Income	Education	MAX(Learner Test Score)	Primary Interest In Course	Hours Coded	How Many Hours A Week Can You Commit To Class	Promise Zone Indicator	H
0	54	11	0	65	0	1	3	0	

```
# Save out base data and labels
def save_to_file(df, file_name):
    with open(SAVE_PATH + file_name, 'w') as f:
        df_out = df.to_csv()
        f.write(df_out)
```

```
save_to_file(df_labels, 'df_labels.csv')
save_to_file(df_valilabels, 'df_valilabels.csv')
save_to_file(df_data, 'df_data.csv')
save_to_file(df_validata, 'df_validata.csv')

df_test = pd.read_csv(DATA_PATH + "df_data.csv", delimiter=',', encoding='utf-8')

pickle.dump( df_labels, open( SAVE_PATH + "df_labels.p", "wb" ) )
pickle.dump( df_valilabels, open( SAVE_PATH + "df_valilabels.p", "wb" ) )

pickle.dump( df_data, open( SAVE_PATH + "df_data.p", "wb" ) )
pickle.dump( df_validata, open( SAVE_PATH + "df_validata.p", "wb" ) )


df_test.head(5)
```

	Unnamed: 0	Age (Birthday Masked)	Income	Education	MAX(Learner Test Score)	Primary Interest In Course	Hours Coded	How Many Hours A Week Can You Commit To Class	In
0	0	54	11	0	65	0	1	3	

Clean and Transform Data

Data is cleaned to replace missing values and normalised

```
def normalise_df(train_df, test_df, AXIS, val_df=1):
    mu = train_df.mean(axis=AXIS)
    sd = train_df.std(axis=AXIS)

    train_df = (train_df - mu) / sd
    test_df = (test_df - mu) / sd
    val_df = (val_df - mu) / sd

    return train_df, test_df, val_df
```

```
def drop_na_df(data_df, labels_df, cols):
    frames = [data_df, labels_df]
    df = pd.concat(frames, axis=1)

    df.dropna(thresh=9, how='all', inplace=True)

    data_df = df[cols]
    labels_df = df['statuscode']

    return data_df, labels_df
```

```
def prepare(column_list, desired_columns, df_data, df_validata, df_labels, df_valilabels,
    df_data = df_data[column_list]
    df_validata = df_validata[column_list]

    df_prep = df_data.copy()
    df_prep_val = df_validata.copy()


    # Split into train/test/validation
    X = df_prep[desired_columns]
    Y = df_labels

    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25,
                                                        random_state=42,
                                                        shuffle=SHUFFLE)

    X_val = df_prep_val[desired_columns]
    y_val = df_valilabels

    # Normalise data
    X_train, X_test, X_val = normalise_df(X_train, X_test, 0, X_val)

    # Ensure values are flattened
    y_train = y_train.values.ravel()
    y_test = y_test.values.ravel()
    y_val = y_val.values.ravel()

    return X_train, X_test, X_val, y_train, y_test, y_val
```

```
SHUFFLE = True
FILL = 0

X_train, X_test, X_val, y_train, y_test, y_val = prepare(
    column_list,desired_columns, df_data, df_validata, df_labels,df_valilabels,
    SHUFFLE, FILL)
```

```
print("This should be 0, 1's:\t", y_train)
```

```
This should be 0, 1's:   [1 1 1 ... 1 0 1]
```

```
np.unique(y_train)
```

```
array([0, 1])
```

Save Cleaned Data

```
# Save out pickled data and labels
pickle.dump( X_train, open( SAVE_PATH + "X_train.p", "wb" ) )
pickle.dump( X_test, open( SAVE_PATH + "X_test.p", "wb" ) )
pickle.dump( X_val, open( SAVE_PATH + "X_val.p", "wb" ) )

pickle.dump( y_train, open( SAVE_PATH + "y_train.p", "wb" ) )
pickle.dump( y_test, open( SAVE_PATH + "y_test.p", "wb" ) )
pickle.dump( y_val, open( SAVE_PATH + "y_val.p", "wb" ) )
```

Checking that the dataframe looks like it should:

```
X_train.head()
```

	Masked)		)		How Many Commit To Class	Indicator	
1341	0.353798	1.832671	0.988618	-0.055229	-2.027239	-0.405866	0.571825
1649	-1.263828	-0.299421	0.018359	0.959409	0.381102	-0.405866	0.571825