Feature processing of the miscellaneous attributes. Need to do things like 1 hot encoding, etc.

https://www.kdnuggets.com/2020/07/easy-guide-data-preprocessing-python.html

https://www.udemy.com/course/machinelearning/learn/lecture/19039248#notes

https://colab.research.google.com/github/google/eng-edu/blob/master/ml/cc/exercises/linear_regression_with_a_real_dataset.ipynb?utm_source=mlcc&utm_campaign=colab-external&utm_medium=referral&utm_content=linear_regression_real_tf2-colab&hl=en

REALLY GOOD: https://www.kaggle.com/dmilla/introduction-to-decision-trees-titanic-dataset#Preparing-the-Titanic-dataset

```
# Imports needed for the script
import numpy as np
import pandas as pd
import re
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls

# For handling "thirteen-> 13"
!pip install word2number
from word2number import w2n
```

```
Collecting word2number
  Downloading https://files.pythonhosted.org/packages/4a/29/a31940c848521f0725f0df6b25dca
Building wheels for collected packages: word2number
  Building wheel for word2number (setup.py) ... done
  Created wheel for word2number: filename=word2number-1.1-cp36-none-any.whl size=5588 sha
  Stored in directory: /root/.cache/pip/wheels/46/2f/53/5f5c1d275492f2fce1cdab9a9bb12d492
Successfully built word2number
Installing collected packages: word2number
Successfully installed word2number-1.1
```

```
# Data paths

MISC_TRAIN = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisio
MISC_TEST = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisions
MISC_EVAL = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisions
```

```
# Loading the data
train = pd.read_csv(MISC_TRAIN)
test = pd.read_csv(MISC_TEST)
eval = pd.read_csv(MISC_EVAL)
```

```
eval = pd.read_csv(NIBE_EVAL)
# features = training_data.values # values returns just the values without column headers etc.
# print(features[24])
train.head(10)

# Merging the data
full_data = [train,test,eval]
full = pd.concat(full_data,keys=['train','test','eval'])
print(full)

# Ok, Now the question.... How to process? Want to use one hot encodings for everything I thin

# Assuming that the order of examples for these coincides with the order
# of examples in the other datasets, thus at some point need to concat my new
# feauture vec here to the other ones...

# Problems..
# Assuming no missing data.....
# Need to convert each column to a vector like [1 0 0 0 ... 0]
# These need to have the same length across each of my sets
# Is this the best way to do this??
# Want to bin the age & num victims - my only two real values
# Victim genders is a list of victims separated by ";"
# i.e. ['not known' 'male' 2 'female;female' 'theft' 'grandLarceny']

# how to deal with these? I guess make a longer vector to handle this? IDK
# These are really interesting probs that I wish we covered in class!
# Maybe I'll leave out victims for now? Or I guess I should prob proccess in a cleaner way
# I'll need to have a one hot encoding that has a vector length the same across
# all my training sets, easy way is to bin... mabe 0 1 2 >2 ?
# Then for victims, maybe I'll say if maj male 0, if maj female 0, not
# necessarily including how many of each
# Age maybe <18 18-30 30-45 45-60 60+ ?
# The rest I think I can do categorical

# Maybe I don't even need to do this and can just adapt my decision tree to deal with multiple
# of binary stuff?

# OR - maybe I make 2 datasets - 1 for decision tree.. and one for the rest.
```

```
             defendant_age defendant_gender  ...  offence_category offence_subcategory
train 0                 62           female  ...             theft      theftFromPlace
      1                 17             male  ...             theft       pocketpicking
      2          not known             male  ...             theft       pocketpicking
      3          not known             male  ...             theft       simpleLarceny
      4                 52             male  ...             theft       pocketpicking
...                    ...              ...  ...               ...                 ...
eval  5245       not known             male  ...             theft      theftFromPlace
      5246       not known             male  ...            sexual              sodomy
      5247       not known             male  ...             theft  stealingFromMaster
      5248              26             male  ...             theft            burglary
      5249              16             male  ...             theft       simpleLarceny

[25000 rows x 6 columns]
```

```
# Ok now, let's do some processing via: https://www.kaggle.com/dmilla/introduction-to-decision-
```

```
# Copy original dataset in case we need it later when digging into interesting features
# WARNING: Beware of actually copying the dataframe instead of just referencing it
# "original_train = train" will create a reference to the train variable (changes in 'train' wi
# train = pd.read_csv(MISC_TRAIN)
# original_train = train.copy() # Using 'copy()' allows to clone the dataset, creating a diffe

# Feature engineering steps taken from Sina and Anisotropic, with minor changes to avoid warnin
# full_data = [train] # will add test and eval here later
dataset = full.copy()
processed_data = []

# Let's go column by column and consider best treatment
# First is age '

# function for helping to process written numbers
def word_to_number(x):
  try:
    return w2n.word_to_num(x)
  except:
    # see if there is any number in phrase and return
    x = str(x)
    number = []
    num = ''
    for i in x:
      if i.isdigit():
        number.append(i)
    if len(number) > 0:
      return int(num.join(number))
    else:
      # Change any other results to NaN
      x = float("nan")
      return x

def victims_processor(x):
  # takes in list of victim genders separated by ;
  # Split on ;
  x = str(x)
  x_split = x.split(';')
  # print(x_split)
  for gender in x_split:
    if gender=='female':
      return 1
  return 0

# AGE
# Convert all ages to numbers
# print(dataset['defendant_age'].unique())
dataset['defendant_age'] = dataset['defendant_age'].apply(word_to_number)
# print(dataset['defendant_age'].unique())

# CUT allows us to select specific bin ranges, which may make sense, maybe
# https://pbpython.com/pandas-qcut-cut.html
# I want all kids under 16 to be own group...
cut_bins = [0,16,25,35,50,110]
dataset['defendant_age'] = pd.cut(dataset['defendant_age'],bins=cut_bins) # set_labels=False to
```

```
dataset['defendant_age'] = pd.cut(dataset['defendant_age'],bins=cut_bins) # Set labels false t
one_hot_age = pd.get_dummies(dataset['defendant_age'],dummy_na=True)

# Now, gender
# print(dataset['defendant_gender'].unique())
# print(dataset['defendant_gender'].value_counts())
one_hot_gender = pd.get_dummies(dataset['defendant_gender'],dummy_na=True)

# Now, number of victims, want to bin these
# print(dataset['num_victims'].value_counts())
victim_bins = [-1,0,1,2,3,4,6,100]
victim_cut = pd.cut(dataset['num_victims'],bins=victim_bins)
# print(victim_cut)
one_hot_num_victims = pd.get_dummies(victim_cut,dummy_na=True)

# Now, victim genders, this will be tough
# Need function to process these, I want to return majority male or female or indeterminate (i
# Actually, let's just return 1 if there is a female in the victim or a 0 if not
victim_gender = dataset['victim_genders'].apply(victims_processor)
# print(victim_gender.value_counts())
# print(dataset['victim_genders'].value_counts())

# Now, offence_category - how to I make sure these are same across all datasets - think maybe
# Then separate at end?
# print(dataset['offence_category'].value_counts())
# print(dataset['offence_subcategory'].value_counts())
one_hot_offence = pd.get_dummies(dataset['offence_category'],dummy_na=True)
one_hot_sub_offence = pd.get_dummies(dataset['offence_subcategory'],dummy_na=True)


processed_df = pd.concat([one_hot_age,one_hot_gender,one_hot_num_victims,victim_gender,one_hot_

print(processed_df.shape)

processed_df.head()
# processed_df.iloc[17498:17510]

# Split sets back apart on key
train_processed = processed_df.loc['train']
test_processed = processed_df.loc['test']
eval_processed = processed_df.loc['eval']

# print(train_processed.shape)
# print(test_processed.shape)
# print(eval_processed.shape)
eval_processed.head()
```

(25000, 84)

	(0.0, 16.0]	(16.0, 25.0]	(25.0, 35.0]	(35.0, 50.0]	(50.0, 110.0]	NaN	female	indeterminate	male	NaN	(-1.0, 0.0]	(0 1
0	0	0	0	0	0	1	0	0	1	0	0	

```
# let's add the misc attributes to the 3 other NLP datasets

# Let's load them all
TRAINING_PATH = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decis
TESTING_PATH = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisi
EVAL_PATH = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisions

TRAINING_PATH_TF = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-de
TESTING_PATH_TF = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-dec
EVAL_PATH_TF = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisi

TRAINING_PATH_BAG = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-
TESTING_PATH_BAG = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-de
EVAL_PATH_BAG = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decis

EVAL_IDS = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisions/


def load_data(file_path,data_to_append,file_name,glove=True):
    with open(file_path) as f:
      raw_data = [line.split() for line in f]
    # I want to convert this to a numpy array
    N = len(raw_data) # Num examples
    D = None
    if glove:
      D = len(raw_data[0]) # num dimensions (of first example...), need to adjust to hardcode r
    else:
      D = 10001 # num dimensions for tfidf
    print("num_examples:",N,"num_dimensions:",D)
    np_data = np.zeros((N,D))
    # np_labels = np.zeros((N,))
    for index,instance in enumerate(raw_data):
      # Store label in numpy array
      label = int(instance[0])
      if label == 0:
        label = -1
      np_data[index][0] = label
      # Store data
      for dim,feat in enumerate(instance[1:]):
        feat_index = int(feat.split(":")[0])
        feat_value = float(feat.split(":")[1])
        np_data[index][feat_index] = feat_value
        # np_data[index][dim] = feat_value

    # Now, convert to pandas df
    df = pd.DataFrame(np_data)
    # Add my data
    df = pd.concat([df,data_to_append],axis=1)
```

```
    # print("labels shape:",np_labels.shape)
    print("intance shape:",df.shape)

    # Save file
    save_path = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decis

    df.to_csv(path_or_buf=save_path,index=False)

    return df


# Glove
# glove_train = load_data(TRAINING_PATH,train_processed,'glove_misc_train')
# load_data(TESTING_PATH,test_processed,'glove_misc_test')
# load_data(EVAL_PATH,eval_processed,'glove_misc_eval')

# TF-IDF
# # tf_train = load_data(TRAINING_PATH_TF,train_processed,'tf_misc_train',False)
# load_data(TESTING_PATH_TF,test_processed,'tf_misc_test',False)
# load_data(EVAL_PATH_TF,eval_processed,'tf_misc_eval',False)

# # BOW
# bow_train = load_data(TRAINING_PATH_BAG,train_processed,'bow_misc_train',False)
# load_data(TESTING_PATH_BAG,test_processed,'bow_misc_test',False)
# load_data(EVAL_PATH_BAG,eval_processed,'bow_misc_eval',False)
```

```
num_examples: 2250 num_dimensions: 10001
intance shape: (2250, 10085)
num_examples: 5250 num_dimensions: 10001
intance shape: (5250, 10085)
num examples: 17500 num dimensions: 10001
```

```
# # Now, append labels to misc attributes, just the attributes themselves
# train_processed.insert(0,'label',training_labels)
# test_processed.insert(0,'label',testing_labels)
# eval_processed.insert(0,'label',eval_labels)

# train_processed.head()

# # Export as csv files
# save_train = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisi
# save_test = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisio
# save_eval = '/content/drive/My Drive/Colab Notebooks/Machine Learning 2020/old-bailey-decisio



# train_processed.to_csv(path_or_buf=save_train,index=False)
# test_processed.to_csv(path_or_buf=save_test,index=False)
# eval_processed.to_csv(path_or_buf=save_eval,index=False)
```

```
# Ok now, let's do some processing via: https://www.kaggle.com/dmilla/introduction-to-decision-

# Copy original dataset in case we need it later when digging into interesting features
# WARNING: Beware of actually copying the dataframe instead of just referencing it
# "original_train = train" will create a reference to the train variable (changes in 'train' wi
train = pd.read_csv(MISC_TRAIN)
original_train = train.copy() # Using 'copy()' allows to clone the dataset, creating a differe

# Feature engineering steps taken from Sina and Anisotropic, with minor changes to avoid warni
full_data = [train] # will add test and eval here later
# full_data = full
processed_data = []

# Let's go column by column and consider best treatment
# First is age '

# function for helping to process written numbers
def word_to_number(x):
  try:
    return w2n.word_to_num(x)
  except:
    # see if there is any number in phrase and return
    x = str(x)
    number = []
    num = ''
    for i in x:
      if i.isdigit():
        number.append(i)
    if len(number) > 0:
      return int(num.join(number))
    else:
      # Change any other results to NaN
      x = float("nan")
```

```
      return x

def victims_processor(x):
  # takes in list of victim genders separated by ;
  # Split on ;
  x = str(x)
  x_split = x.split(';')
  # print(x_split)
  for gender in x_split:
    if gender=='female':
      return 1
  return 0

# AGE
for idx,dataset in enumerate(full_data):
    # Convert all ages to numbers
    # print(dataset['defendant_age'].unique())
    dataset['defendant_age'] = dataset['defendant_age'].apply(word_to_number)
    # print(dataset['defendant_age'].unique())

    # Print average and std of age
    # age_avg = dataset['defendant_age'].mean()
    # age_std = dataset['defendant_age'].std()
    # print("average age:",age_avg,"age std:",age_std)
    # print(dataset['defendant_age'].describe())

    # Now I want to bin these values
    # Mapping Age
    # dataset.loc[ dataset['defendant_age'].isna(), 'defendant_age'] = 0
    # dataset.loc[(dataset['defendant_age'] > 0) & (dataset['defendant_age'] <= 16), 'defendant
    # dataset.loc[(dataset['defendant_age'] > 16) & (dataset['defendant_age'] <= 32), 'defendar
    # dataset.loc[(dataset['defendant_age'] > 32) & (dataset['defendant_age'] <= 48), 'defendar
    # dataset.loc[(dataset['defendant_age'] > 48) & (dataset['defendant_age'] <= 64), 'defendar
    # dataset.loc[ dataset['defendant_age'] > 64, 'defendant_age'] = 5
    # dataset = dataset.astype({'defendant_age': 'int32'}) # need to assign to new variable
    # print(dataset['defendant_age'].head(5))

    # Can potentially do this better with "category" data type... or qcut
    # bins = 4
    # QCUT automatically orginizes bins
    # View bin ranges
    # dataset = dataset.astype({'defendant_age' : 'category'})
    # my_bins = pd.qcut(dataset['defendant_age'],
    #                         q=bins,
    #                         precision=0)
    # print(my_bins.value_counts())
    # dataset['defendant_age'] = pd.qcut(dataset['defendant_age'],
    #                         q=bins,
    #                         labels=False,
    #                         precision=0)
    # CUT allows us to select specific bin ranges, which may make sense, maybe
    # https://pbpython.com/pandas-qcut-cut.html
    # I want all kids under 16 to be own group...
    cut_bins = [0,16,25,35,50,110]
    dataset['defendant_age'] = pd.cut(dataset['defendant_age'],bins=cut_bins) # set labels=Fals
```

```
    # cut_labels = pd.cut(dataset['defendant_age'],bins=cut_bins)
    # then, set NaN's, or can specify in get dummys
    # dataset.loc[ dataset['defendant_age'].isna(), 'defendant_age'] = len(cut_bins)-1
    # dataset = dataset.astype({'defendant_age' : 'int64'})
    # print(dataset['defendant_age'].head(10))
    # print(dataset['defendant_age'].value_counts())
    # print(dataset.head())
    # Now convert to one hot encodiing
    # get_dummies
    # Pass ranges as labels?
    one_hot_age = pd.get_dummies(dataset['defendant_age'],dummy_na=True)


    # Now, gender
    # print(dataset['defendant_gender'].unique())
    # print(dataset['defendant_gender'].value_counts())
    one_hot_gender = pd.get_dummies(dataset['defendant_gender'])

    # Now, number of victims, want to bin these
    # print(dataset['num_victims'].value_counts())
    victim_bins = [-1,0,1,2,3,4,6,100]
    victim_cut = pd.cut(dataset['num_victims'],bins=victim_bins)
    # print(victim_cut)
    one_hot_num_victims = pd.get_dummies(victim_cut,dummy_na=True)

    # Now, victim genders, this will be tough
    # Need function to process these, I want to return majority male or female or indeterminate
    # Actually, let's just return 1 if there is a female in the victim or a 0 if not
    victim_gender = dataset['victim_genders'].apply(victims_processor)
    # print(victim_gender.value_counts())
    # print(dataset['victim_genders'].value_counts())

    # Now, offence_category - how to I make sure these are same across all datasets - think may
    # Then separate at end?
    print(dataset['offence_category'].value_counts())
    print(dataset['offence_subcategory'].value_counts())

    processed_df = pd.concat([one_hot_age,one_hot_gender,one_hot_num_victims,victim_gender],axi
    # Assign dataset
    processed_data.append(processed_df)

# for dataset in full_data:
#   print(dataset['defendant_gender'].unique())
# print(train.head(10))
train_processed = processed_data[0]
train_processed.head(10)
```

```
theft            12387
deception         1248
royalOffences      787
breakingPeace      728
sexual             709
violentTheft       629
kill               620
miscellaneous      284
damage             108
Name: offence_category, dtype: int64
grandLarceny                    4020
simpleLarceny                   2283
theftFromPlace                  1219
pocketpicking                   1182
stealingFromMaster               869
coiningOffences                  740
burglary                         622
other                            547
wounding                         530
forgery                          510
embezzlement                     496
fraud                            477
animalTheft                      369
highwayRobbery                   321
robbery                          308
housebreaking                    279
manslaughter                     271
murder                           258
shoplifting                      255
rape                             240
bigamy                           220
receiving                        212
perjury                          206
mail                             122
pettyLarceny                     104
assault                           94
sodomy                            88
assaultWithIntent                 68
arson                             67
concealingABirth                  66
libel                             65
indecentAssault                   61
infanticide                       57
bankrupcy                         48
returnFromTransportation          40
extortion                         36
kidnapping                        20
pervertingJustice                 19
taxOffences                       18


seditiousWords                    10
```

```
# ENCODING CATEGORICAL DATA - I'm using scikit learn here.... hope that's ok, double check
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough
features = np.array(ct.fit_transform(features))
```

0	0	0	0	0	1	0	1	0	0	0	1
1	0	1	0	0	0	0	0	0	1	0	1
2	0	0	0	0	0	1	0	0	1	0	1
3	0	0	0	0	0	1	0	0	1	0	1
4	0	0	0	0	1	0	0	0	1	0	1
5	0	0	0	1	0	0	0	0	1	1	0
6	0	0	0	0	0	1	0	0	1	0	1
7	0	0	0	0	0	1	1	0	0	0	1
8	0	0	1	0	0	0	0	0	1	1	0
9	0	1	0	0	0	0	0	0	1	0	1