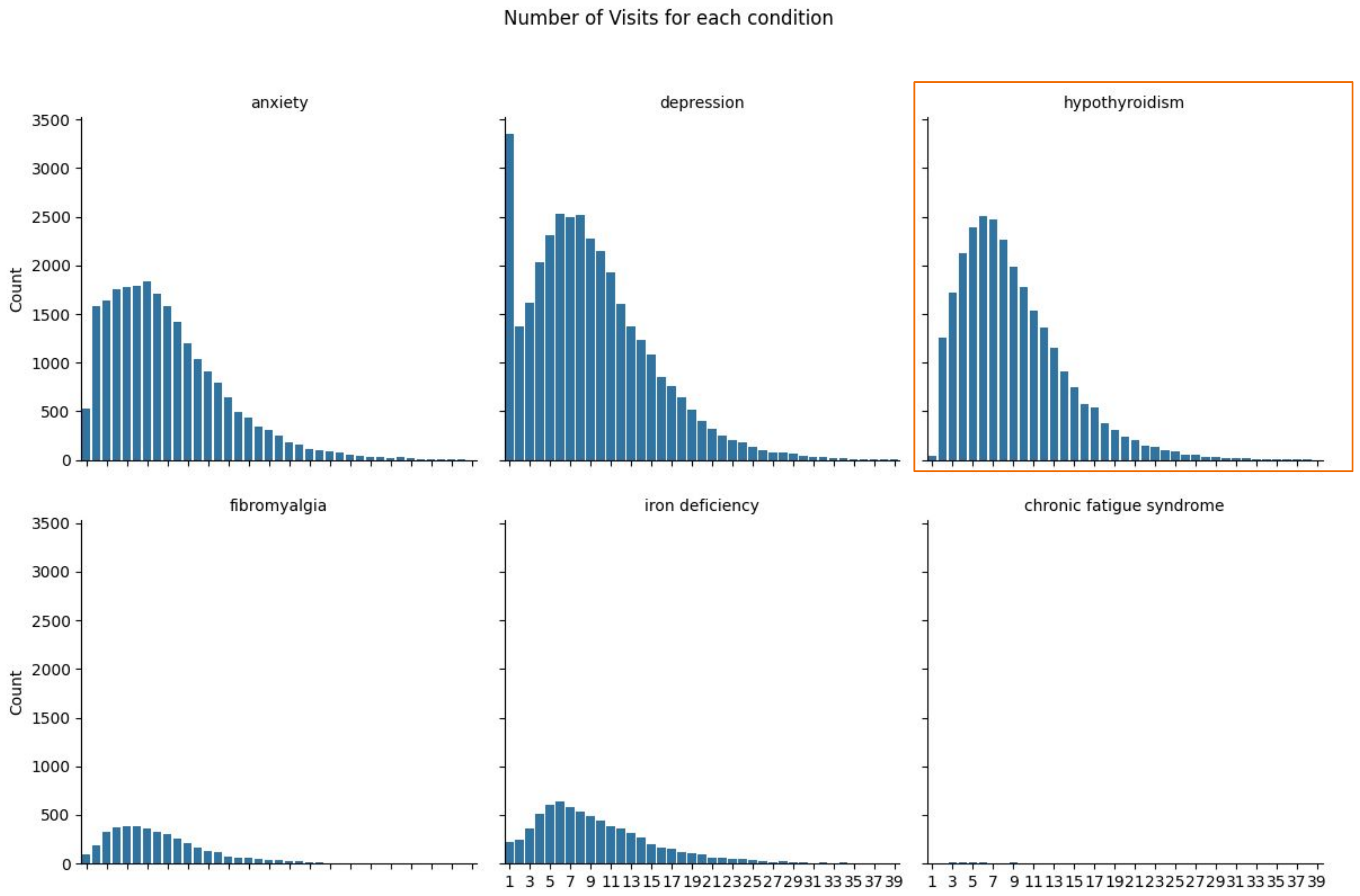
Diagnosing Hypothyroidism

Hypothyroidism is one of the top 10 misdiagnosed diseases in America

Thyroid Disease is number 6 on the list
https://ml-law.net/medical-errors-misdiagnosis-negligence/10-most-commonly-medical-misdiagnosed-illnesses



Very few case of hypothyroidism are diagnosed with only one visit compared to some other disease especially depression.

Symptoms

- Fatigue
- Weight gain
- Depression
- Constipation
- A Sudden Spike in Cholesterol:
  (High cholesterol is sometimes the only evidence of an underactive thyroid in an older person. Because this sign may stand alone, high cholesterol warrants a thyroid evaluation.)

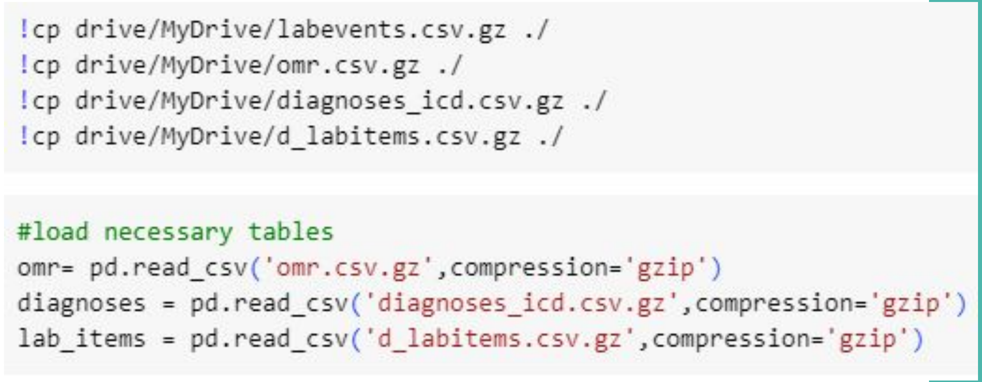
https://www.centrichealth.ie/health-wellness-blog/hypothyroidism-and-dementia-misdiagnosis-as-we-age/

Hypothyroidism is often mistaken for:

- Anxiety
- Bipolar disorder-cyclothymia
- Iron deficiency
- Chronic fatigue syndrome
- Fibromyalgia
- Depression

MIMIC IV

Tables we choose



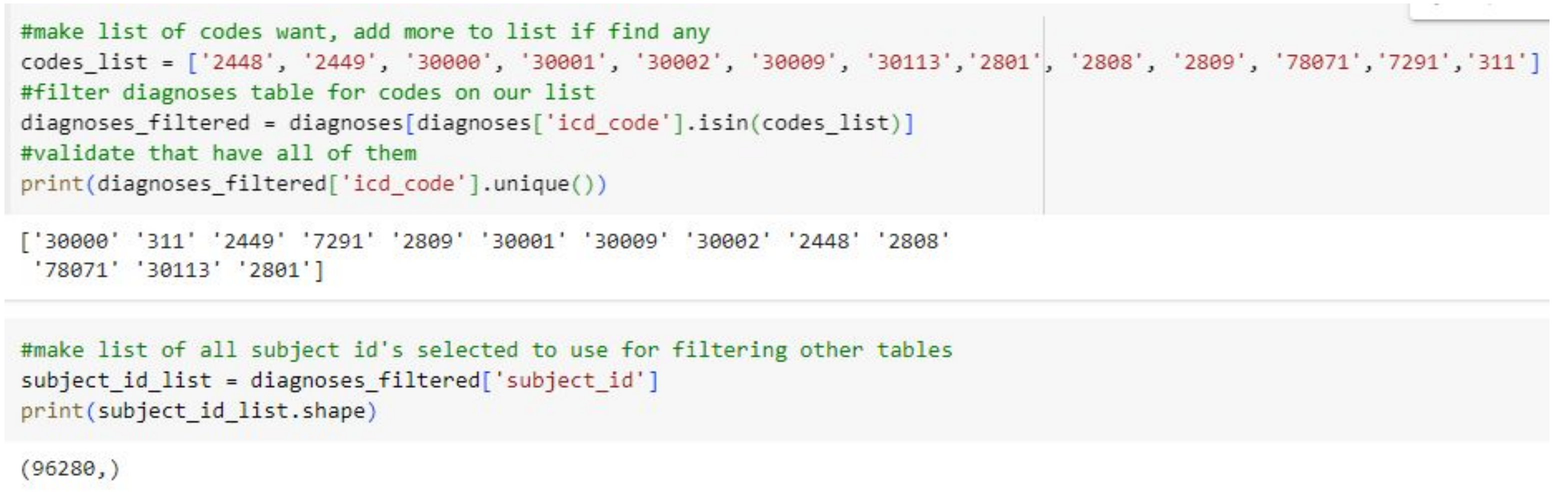
```
!cp drive/MyDrive/labevents.csv.gz ./
!cp drive/MyDrive/omr.csv.gz ./
!cp drive/MyDrive/diagnoses_icd.csv.gz ./
!cp drive/MyDrive/d_labitems.csv.gz ./
```

```
#load necessary tables
omr= pd.read_csv('omr.csv.gz',compression='gzip')
diagnoses = pd.read_csv('diagnoses_icd.csv.gz',compression='gzip')
lab_items = pd.read_csv('d_labitems.csv.gz',compression='gzip')
```

Omr: contains height and weight, bmi, blood pressure and eGFR. - Weight gain, high blood pressure and kidney function can all be indicators of hypothyroidism.

Diagnoses_icd: to see who is actually diagnosed with hypothyroidism.

Lab_items: see if we can use certain labs to indicate hypothyroidism.



```
#make list of codes want, add more to list if find any
codes_list = ['2448', '2449', '30000', '30001', '30002', '30009', '30113','2801', '2808', '2809', '78071','7291','311']
#filter diagnoses table for codes on our list
diagnoses_filtered = diagnoses[diagnoses['icd_code'].isin(codes_list)]
#validate that have all of them
print(diagnoses_filtered['icd_code'].unique())
```

```
['30000' '311' '2449' '7291' '2809' '30001' '30009' '30002' '2448' '2808'
 '78071' '30113' '2801']
```

```
#make list of all subject id's selected to use for filtering other tables
subject_id_list = diagnoses_filtered['subject_id']
print(subject_id_list.shape)
```

```
(96280,)
```

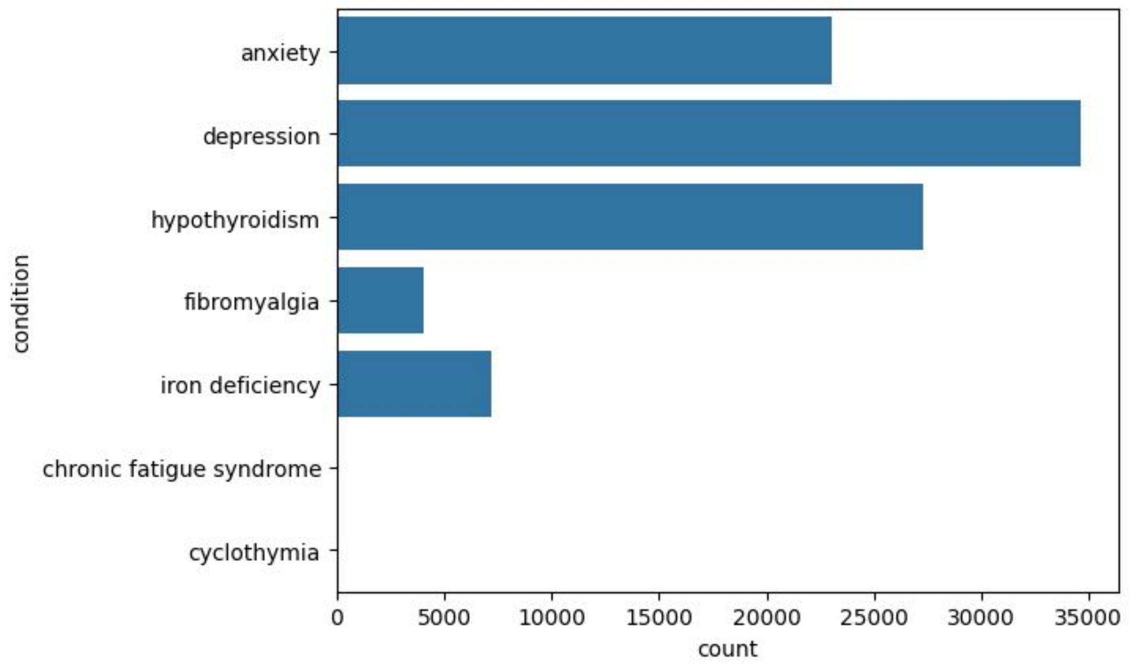
Filtering

Selecting only the ones for hypothyroidism and the few disease that it’s often mistaken for.

How often does each condition appear?

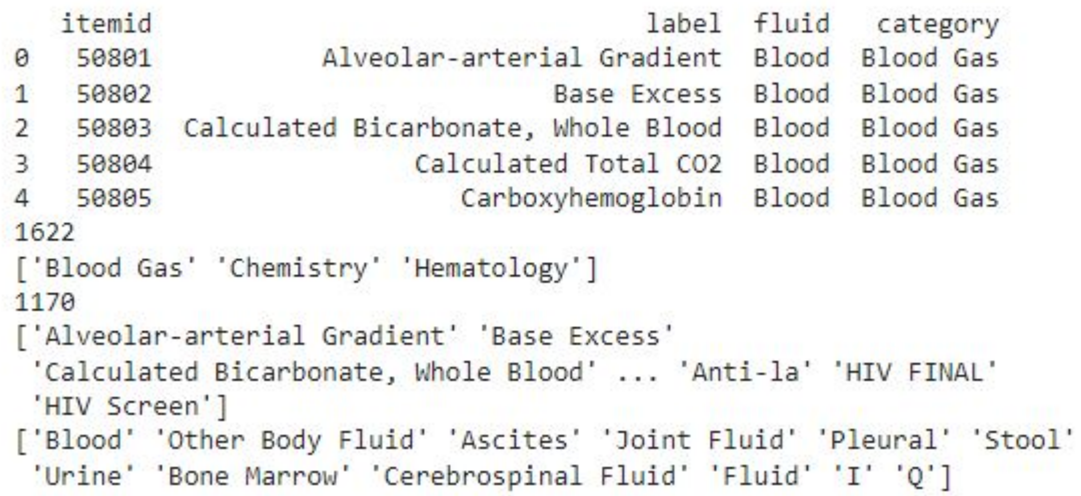
```
sns.countplot(diagnoses_filtered, y= 'condition')
plt.show()
```

```
#group by each condition
diagnoses_filtered.loc[diagnoses_filtered['icd_code'].isin(['2448', '2449']),'condition'] = 'hypothyroidism'
diagnoses_filtered.loc[diagnoses_filtered['icd_code'].isin(['30000', '30001', '30002', '30009']),'condition'] = 'anxiety'
diagnoses_filtered.loc[diagnoses_filtered['icd_code'].isin(['30113']),'condition'] = 'cyclothymia'
diagnoses_filtered.loc[diagnoses_filtered['icd_code'].isin(['2801', '2808', '2809']),'condition'] = 'iron deficiency'
diagnoses_filtered.loc[diagnoses_filtered['icd_code'].isin(['78071']),'condition'] = 'chronic fatigue syndrome'
diagnoses_filtered.loc[diagnoses_filtered['icd_code'].isin(['7291']),'condition'] = 'fibromyalgia'
diagnoses_filtered.loc[diagnoses_filtered['icd_code'].isin(['311']),'condition'] = 'depression'
print(diagnoses_filtered['condition'].unique())
print(diagnoses_filtered.head())
```



Labs item table is only a text description for each item id. We need the labevents table.

```
#check what lab items table
looks like
print(lab_items.head())
print(len(lab_items))
print(lab_items['category'].uniq
ue())
print(lab_items['label'].nunique
())
print(lab_items['label'].unique(
))
print(lab_items['fluid'].unique(
))
```

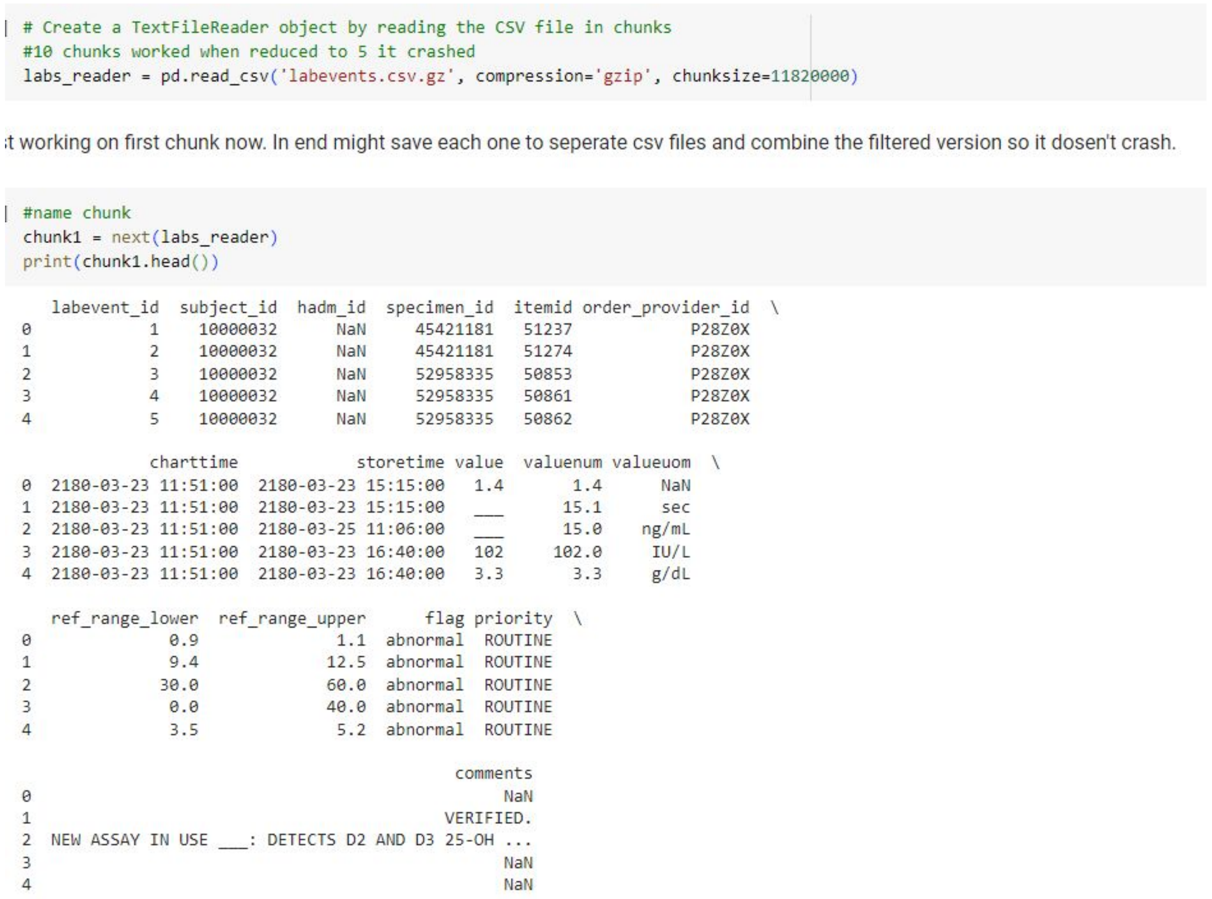


```
   itemid                              label  fluid   category
0   50801           Alveolar-arterial Gradient  Blood  Blood Gas
1   50802                         Base Excess  Blood  Blood Gas
2   50803  Calculated Bicarbonate, Whole Blood  Blood  Blood Gas
3   50804                  Calculated Total CO2  Blood  Blood Gas
4   50805                    Carboxyhemoglobin  Blood  Blood Gas
1622
['Blood Gas' 'Chemistry' 'Hematology']
1170
['Alveolar-arterial Gradient' 'Base Excess'
 'Calculated Bicarbonate, Whole Blood' ... 'Anti-la' 'HIV FINAL'
 'HIV Screen']
['Blood' 'Other Body Fluid' 'Ascites' 'Joint Fluid' 'Pleural' 'Stool'
 'Urine' 'Bone Marrow' 'Cerebrospinal Fluid' 'Fluid' 'I' 'Q']
```

Loading lab events table.

The lab events table is very large and the notebook kept on crashing as there was not enough ram available. We experimented using different chunk sizes and 10 chunks worked.

When we further reduced it to 5 it crashed again.



```
# Create a TextFileReader object by reading the CSV file in chunks
#10 chunks worked when reduced to 5 it crashed
labs_reader = pd.read_csv('labevents.csv.gz', compression='gzip', chunksize=11820000)
```

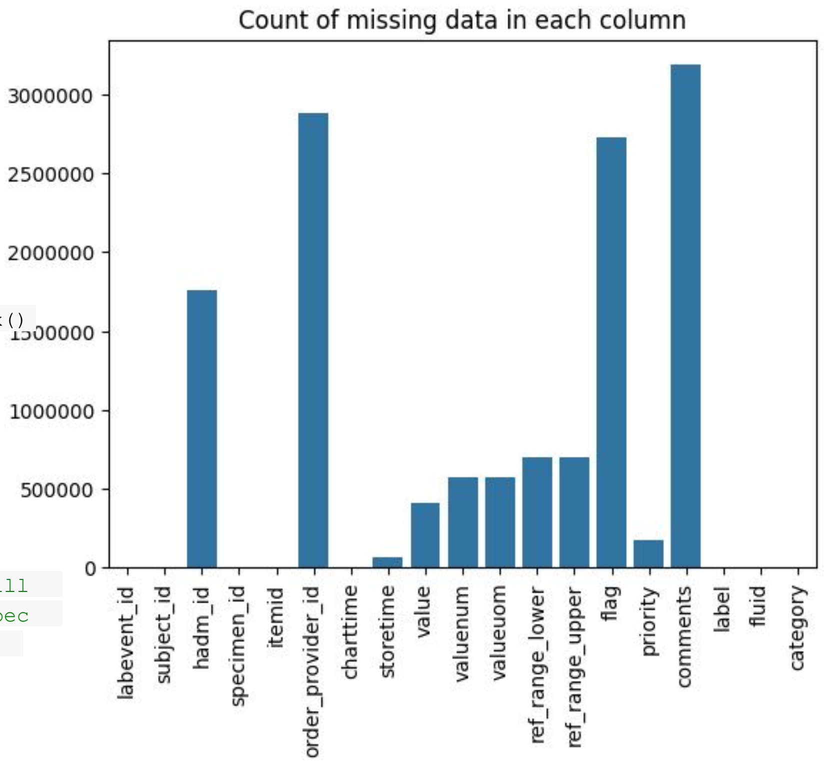
t working on first chunk now. In end might save each one to seperate csv files and combine the filtered version so it dosen't crash.

```
#name chunk
chunk1 = next(labs_reader)
print(chunk1.head())
```

```
   labevent_id  subject_id  hadm_id  specimen_id  itemid order_provider_id  \
0            1    10000032      NaN     45421181   51237            P28Z0X
1            2    10000032      NaN     45421181   51274            P28Z0X
2            3    10000032      NaN     52958335   50853            P28Z0X
3            4    10000032      NaN     52958335   50861            P28Z0X
4            5    10000032      NaN     52958335   50862            P28Z0X

             charttime            storetime value  valuenum valueuom  \
0  2180-03-23 11:51:00  2180-03-23 15:15:00   1.4       1.4      NaN
1  2180-03-23 11:51:00  2180-03-23 15:15:00   ___      15.1      sec
2  2180-03-23 11:51:00  2180-03-25 11:06:00   ___      15.0    ng/mL
3  2180-03-23 11:51:00  2180-03-23 16:40:00   102     102.0     IU/L
4  2180-03-23 11:51:00  2180-03-23 16:40:00   3.3       3.3     g/dL

   ref_range_lower  ref_range_upper      flag priority  \
0              0.9              1.1  abnormal  ROUTINE
1              9.4             12.5  abnormal  ROUTINE
2             30.0             60.0  abnormal  ROUTINE
3              0.0             40.0  abnormal  ROUTINE
4              3.5              5.2  abnormal  ROUTINE

                                            comments
0                                                NaN
1                                          VERIFIED.
2  NEW ASSAY IN USE ___: DETECTS D2 AND D3 25-OH ...
3                                                NaN
4                                                NaN
```

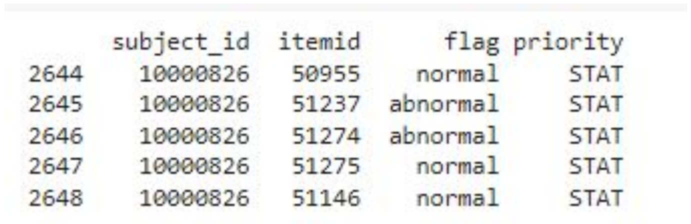
Null values

```
#filter first_chunk_merg for patients from diagnosed table
labs_chunk1_merg_filter =
first_chunk_merg[first_chunk_merg[ 'subject_id' ].isin(subject_id_list)]
#check if worked and is reduced
print(labs_chunk1_merg_filter.shape)
print(first_chunk_merg.shape)
print(labs_chunk1_merg_filter.shape)
print(labs_chunk1_merg_filter.isna(). sum())
data_na = pd.DataFrame(labs_chunk1_merg_filter.isna().  sum()).reset_index()
#plot missing data
sns.barplot(data=data_na, x= 'index',y=0)
plt.ticklabel_format(style= 'plain', axis='y')
plt.xticks(rotation= 90)
plt.title('Count of missing data in each column'  )
plt.ylabel("")
plt.xlabel("")
plt.show()
#remove hadm_id and order_provider_id and comments too many null values will
also drop timings for now.also remove ref range lower and upper for now. bec
seems like if not in range it's flagged as abnormal go off that value for
now.
labs_chunk1_merg_filter
=labs_chunk1_merg_filter.drop([ 'hadm_id','order_provider_id' ,
                                                       'comments',
'charttime','storetime','ref_range_lower' ,'ref_range_upper' ],axis = 1)
```



Value column also has text which is hard to run models on. The value num is only the ones with integers. Values that are not within normal range are flagged. For now will just work with that and use all values that flagged as abnormal.

```
labs_chunk1_merg_filter
=labs_chunk1_merg_filter.drop(['valuenum',
'valueuom','label' 'fluid','category',
'labevent_id', 'specimen_id','value'
 ],axis = 1)
print(labs_chunk1_merg_filter.head())
```



```
      subject_id  itemid      flag priority
2644    10000826   50955    normal     STAT
2645    10000826   51237  abnormal     STAT
2646    10000826   51274  abnormal     STAT
2647    10000826   51275    normal     STAT
2648    10000826   51146    normal     STAT
```

Filter to only use the ones that are flagged as abnormal. Once only have abnormal ones can remove that column and changed priority column to 1 if it was marked as stat.

```
#filter to only the ones which are abnormal and try to see which tests
with abnormal reslults can predict hypothrodism
labs_chunk1_merg_filter =
labs_chunk1_merg_filter[labs_chunk1_merg_filter['flag'] == 'abnormal']
print(labs_chunk1_merg_filter.shape)
#drop flag column since only have abnormal now
labs_chunk1_merg_filter =labs_chunk1_merg_filter.drop(['flag'],axis = 1)
#turn priority column stat into 1
# Define the mapping dictionary
priority_mapping = {'STAT': 1, 'ROUTINE': 0}

# Replace values using the mapping dictionary
labs_chunk1_merg_filter['priority'] =
labs_chunk1_merg_filter['priority'].replace(priority_mapping)

# Print unique values to verify the transformation
print(labs_chunk1_merg_filter['priority'].unique())

#change name to priority_stat
labs_chunk1_merg_filter =
labs_chunk1_merg_filter.rename(columns={'priority': 'priority_stat'})

# Print the updated DataFrame to verify the change
print(labs_chunk1_merg_filter.head())
```

Filter to only include top 20 tests that have abnormal results and that person was diagnosed with hypothyroidism

'INR(PT)' 'PT' 'Hematocrit' 'Hemoglobin' 'MCH' 'MCHC' 'MCV' 'Neutrophils' 'Platelet Count' 'RDW' 'Red Blood Cells' 'White Blood Cells' 'Bicarbonate' 'Chloride' 'Creatinine' 'Glucose' 'Urea Nitrogen' 'Calcium, Total' 'Phosphate' 'RDW-SD'

Get top 20

```
# Initialize an empty list to store the mode for 'hypothyroidism'
for each round
mode_list = []
hypo_mode_list = []

# Loop for 20 rounds
for i in range(1, 21):

    # Group by 'condition' and find the mode of 'itemid' for each
condition
    stats =
merged_labdg.groupby('condition')['itemid'].agg(mode=lambda x:
x.mode().iloc[0])

    # Append the mode for 'hypothyroidism' to the list
    stats.reset_index(inplace=True)
    hypothyroidism_mode = stats.iloc[5,1]
    mode_list.append(stats)
    hypo_mode_list.append(hypothyroidism_mode)

    # Remove rows where 'itemid' is equal to the mode for
'hypothyroidism'
    merged_labdg = merged_labdg[merged_labdg['itemid'] !=
hypothyroidism_mode]


# Print the DataFrame
print(hypo_mode_list)
```

Split testing and training

Had to split subject id first to ensure each person stays in same group.

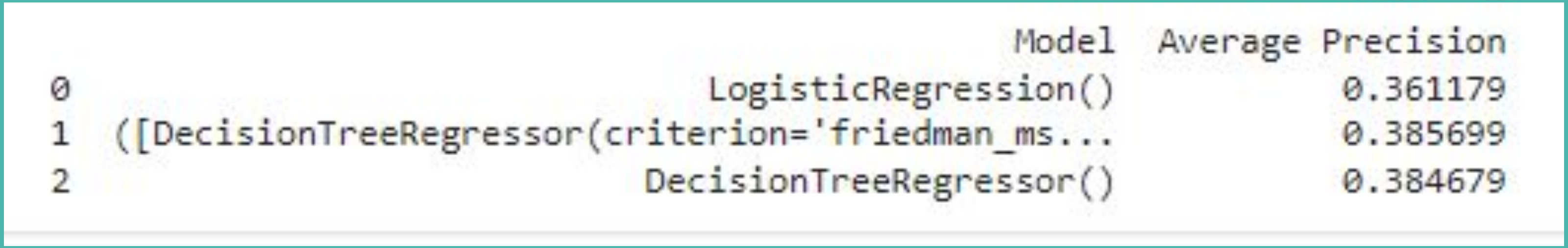
```
from sklearn.model_selection import train_test_split
# Split id's into training and testing sets for when
split data
X_trainid, X_testid = train_test_split(subject_id_list,
test_size=0.25, random_state=42)
# Print the shapes of the training and testing sets
print(X_trainid.shape)
print(X_testid.shape)
#split merged_labdg according to id to ensure that each
person stays in one group
X_train_labs =
merged_labdg[merged_labdg['subject_id'].isin(X_trainid)]
X_test_labs =
merged_labdg[~merged_labdg['subject_id'].isin(X_trainid)]
print(X_train_labs.shape)
print(X_test_labs.shape)
#divide each into x and y. Y being condition
y_train_omr = X_train_omr['condition']
y_test_omr = X_test_omr['condition']
y_train_labs = X_train_labs['condition']
y_test_labs = X_test_labs['condition']
```

Models

Created a function to run the models.
Included logistic regression, gradient boosting, and decision tree

```
def evaluate_models (X_train, X_test, y_train, y_test):
    #import models
    #logistic regression
    from sklearn.linear_model  import  LogisticRegression
    #gradient boosting
    from sklearn.ensemble  import  GradientBoostingRegressor
    # decision tree
    from sklearn.tree  import  DecisionTreeRegressor
    from sklearn.metrics  import  average_precision_score # try different models

    #list of models

models=[LogisticRegression(),GradientBoostingRegressor(),DecisionTreeRegressor()
]
    #empty list to store avg precision
    avg_precisions = []

    #run models
    for model  in  models:

      #fit the model
      model.fit(X_train,y_train_labs)
      y_pred = model.predict(X_test)
      # Calculate the average precision of the model
      avg_precision = average_precision_score(y_test, y_pred)

      # Append the average precision to the list
      avg_precisions.append(avg_precision)

    # Create a dataframe of the average precision of each model
    df_avg_precisions = pd.DataFrame({  'Model' : models,  'Average Precision' :
avg_precisions})

    # Print the dataframe
    print(df_avg_precisions)
```

Model Results


```
                                              Model  Average Precision
0                              LogisticRegression()           0.361179
1  ([DecisionTreeRegressor(criterion='friedman_ms...           0.385699
2                           DecisionTreeRegressor()           0.384679
```

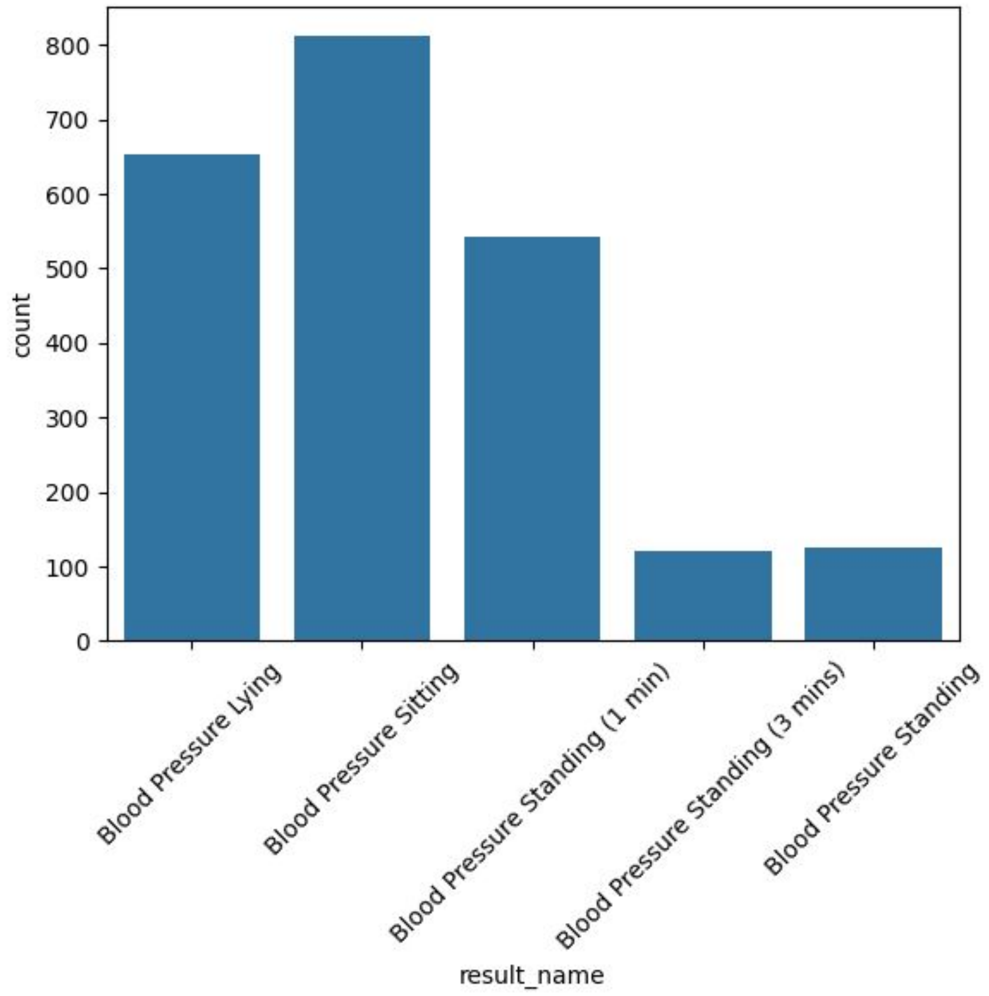
Gradient boosting model worked best at .385 avg precision

- Next steps:(need more time for these)
- Would want to optimize model to get it to perform better
- Also had omr data that didn’t manage to merge yet or run model on. But we had started getting the data ready.
- Would use omr to figure out it have weight gain as well and use that metric.

Omr table

Has few values for blood pressure as well as height and weight consolidated all that.

```
#consolidate all blood pressure
values
omr_filtered.loc[omr_filtered['resu
lt_name'].isin([ 'Blood Pressure
Sitting', 'Blood Pressure Standing
(1 min)',
 'Blood Pressure Lying', 'Blood
Pressure Standing (3 mins)',
 'Blood Pressure Standing']),
'result_name'] = 'Blood Pressure'
print(omr_filtered['result_name'].u
nique())
```


count
Blood Pressure Lying
Blood Pressure Sitting
Blood Pressure Standing (1 min)
Blood Pressure Standing (3 mins)
Blood Pressure Standing
result_name