Team9_Project1

November 5, 2025

[400]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

df_23 = pd.read_csv(
    "/mnt/c/Users/yingz/Documents/BIOINFO575/Project_option_1/
 ↪23andme_v5_hg19_ref.txt/23andme_v5_hg19_ref.txt",

    sep = "\t", # separat columns by tab characters

    comment = "#", # ignore lines starting with "#" as comments

    header = None, # exclude header row in the data

    names = ["CHR", "POS", "dbSNP_ID", "ALLELE"], # assign these column names␣
 ↪to the 4 fields

    dtype = {"CHR":"string","POS":"int64","dbSNP_ID":"string","ALLELE":
 ↪"string"} # enforce correct data types for each column
)

df_23
```

[400]:
```
          CHR      POS     dbSNP_ID ALLELE
0        chr1    69869  rs548049170      T
1        chr1    74792   rs13328684      G
2        chr1   565508    rs9283150      G
3        chr1   726912      i713426      A
4        chr1   727841  rs116587930      G
…         …       …            …      …
638458   chrM    16524     i4000693      A
```

```
638459  chrM   16524      i704756      A
638460  chrM   16525      i705255      A
638461  chrM   16526     i4000757      G
638462  chrM   16526      i701671      G

[638463 rows x 4 columns]
```

```
[401]: phrase = "15-year cumulative" # define the exact phrase to search for in the␣
        ↪file
       hits = [] # initialize an empty list that will store the 1-based line numbers␣
        ↪containing the phrase

       with open("/mnt/c/Users/yingz/Documents/BIOINFO575/Project_option_1/
        ↪variantAnnotations/var_drug_ann.tsv",
                 "r", encoding="utf-8", errors="ignore") as f: # open the TSV file for␣
        ↪reading as UTF-8 text while ignoring decoding errors
           line_number = 0  # start a manual counter for line numbers
           for line in f:  # iterate over each line in the file one by one
               line_number += 1  # increase the counter by one for each line
               if phrase in line:  # check whether the current line contains the␣
        ↪target phrase
                   hits.append(line_number) # record the current line number if the␣
        ↪phrase is present

       print("Lines containing the phrase:", hits) # I have manually revise the bad␣
        ↪line, because I think directly skip a data line is unreasonable.

       var_df = pd.read_csv(
           "/mnt/c/Users/yingz/Documents/BIOINFO575/Project_option_1/
        ↪variantAnnotations/var_drug_ann.tsv",
           sep = "\t", # read the file as a tab-separated table
           dtype = "string" # keep all columns as string type to avoid unintended type␣
        ↪coercion
       )

       var_df.head()
```

```
Lines containing the phrase: [1460]
```

```
[401]:   Variant Annotation ID   Variant/Haplotypes    Gene  \
       0            1451834452  CYP3A4*1, CYP3A4*17  CYP3A4
       1            1453073660            rs2909451    DPP4
       2            1453071582             rs706795   FAIM2
       3            1451138786           rs16918842   OPRK1
       4            1448257202   CYP2C9*1, CYP2C9*3  CYP2C9

                                                 Drug(s)      PMID  \
0                                      nifedipine  15634941
1                                     sitagliptin  39792745
2  citalopram, escitalopram, fluoxetine, fluvoxam…  40054571
3                                          heroin  31940240
4                                        warfarin  27488176

    Phenotype Category Significance  \
0  Other, Metabolism/PK   not stated
1              Efficacy         yes
2              Efficacy          no
3                Dosage          no
4                Dosage         yes

                                             Notes  \
0  in vitro expression of the recombinant CYP3A4*…
1  "Patients with the rs2909451 TT genotype in th…
2  "We observed nominally significant association…
3  No significant difference in allele or genotyp…
4                                             <NA>

                                          Sentence Alleles  … isPlural  \
0  CYP3A4 *17 is associated with decreased metabo…     *17  …       Is
1  Genotype TT is associated with decreased respo…      TT  …       Is
2  Allele T is associated with increased response…       T  …       Is
3  Allele T is not associated with dose of heroin…       T  …       Is
4  CYP2C9 *3 is associated with decreased dose of…      *3  …       Is

  Is/Is Not associated Direction of effect    PD/PK terms  \
0      Associated with           decreased  metabolism of
1      Associated with           decreased    response to
2      Associated with           increased    response to
3  Not associated with                <NA>        dose of
4      Associated with           decreased        dose of

  Multiple drugs And/or Population types    Population Phenotypes or diseases  \
0                  <NA>             <NA>                                 <NA>
1                  <NA>   in people with      Other:Diabetes Mellitus, Type 2
2                    or   in people with  Other:Obsessive-Compulsive Disorder
3                  <NA>   in people with             Other:Heroin Dependence
4                  <NA>             <NA>                                 <NA>

  Multiple phenotypes or diseases And/or Comparison Allele(s) or Genotype(s)  \
0                                  <NA>                                  *1
1                                  <NA>                                <NA>
2                                  <NA>                                   C
3                                  <NA>                                   C
4                                  <NA>                               *1/*1
```

```
  Comparison Metabolizer types
0                         <NA>
1                         <NA>
2                         <NA>
3                         <NA>
4                         <NA>

[5 rows x 22 columns]
```

[402]:
```
# Merge
rs_lists = var_df["Variant/Haplotypes"].str.findall(r"rs[0-9]+") # extract all␣
 ↪rsIDs from the free-text column as a list per row
var_long = var_df.loc[rs_lists.str.len() > 0].copy() # keep only rows that␣
 ↪contain at least one rsID and make a writable copy
var_long["dbSNP_ID"] = rs_lists # attach the list of rsIDs to a new column␣
 ↪named dbSNP_ID
var_long = var_long.explode("dbSNP_ID") # convert each list of rsIDs into␣
 ↪multiple rows so that there is one rsID per row

ph = var_long[[ # select the columns that are needed for the phenotype-drug-
variant table
    "dbSNP_ID", "Gene", "Drug(s)", "PMID",
    "Phenotype Category", "Significance", "Notes", "Sentence", "Alleles"
]].rename(columns={ # standardize column names to consistent identifiers
    "Gene": "GENE_SYMBOL",
    "Drug(s)": "DRUG_NAME",
    "Phenotype Category": "PHENOTYPE_CATEGORY",
    "Significance": "SIGNIFICANCE",
    "Notes": "NOTES",
    "Sentence": "SENTENCE",
    "Alleles": "ALLELE_PharmGKB"
})

# keep rsID and allele from the 23andMe reference and rename the allele column

g23 = df_23[["dbSNP_ID", "ALLELE"]].rename(columns = {"ALLELE":␣
 ↪"ALLELE_23andme"})

# join PharmGKB rows with 23andMe alleles on matching rsID and retain only␣
 ↪matches

merged = ph.merge(g23, on = "dbSNP_ID", how = "inner")

merged.head()
```

```
[402]:      dbSNP_ID GENE_SYMBOL                                         DRUG_NAME  \
       0    rs706795       FAIM2  citalopram, escitalopram, fluoxetine, fluvoxam…
       1  rs16918842       OPRK1                                            heroin
       2    rs163184       KCNQ1                                       sitagliptin
       3   rs7754840      CDKAL1                                       sitagliptin
       4   rs1799853      CYP2C9                                       sitagliptin

              PMID PHENOTYPE_CATEGORY SIGNIFICANCE  \
       0  40054571           Efficacy           no
       1  31940240             Dosage           no
       2  39792745           Efficacy          yes
       3  39792745           Efficacy           no
       4  39792745           Efficacy          yes

                                                      NOTES  \
       0  "We observed nominally significant association…
       1  No significant difference in allele or genotyp…
       2  "KCNQ1 gene polymorphisms also significantly a…
       3  "Patients with the rs7754840 CG genotype showe…
       4  "CYP2C9 gene polymorphisms also significantly …

                                                   SENTENCE ALLELE_PharmGKB  \
       0  Allele T is associated with increased response…               T
       1  Allele T is not associated with dose of heroin…               T
       2  Genotype GG is associated with decreased respo…              GG
       3  Genotype CG is associated with increased respo…              CG
       4  Genotype TT is associated with decreased respo…              TT

         ALLELE_23andme
       0              T
       1              C
       2              T
       3              G
       4              C
```

```
[403]: # create a subset that retains only rows with SIGNIFICANCE equal to "yes" and
        ↪PHENOTYPE_CATEGORY equal to "Efficacy"

       eff_sig = merged[(merged["SIGNIFICANCE"] == "yes") &
        ↪(merged["PHENOTYPE_CATEGORY"] == "Efficacy")]
       eff_sig.head()
```

```
[403]:     dbSNP_ID GENE_SYMBOL                                         DRUG_NAME  \
       2    rs163184       KCNQ1                                       sitagliptin
       4   rs1799853      CYP2C9                                       sitagliptin
       5   rs7903146      TCF7L2                                         exenatide
       8   rs8099917       IFNL3  peginterferon alfa-2a, peginterferon alfa-2b, …
```

```
9  rs8099917       IFNL3                  peginterferon alfa-2b, ribavirin

       PMID PHENOTYPE_CATEGORY SIGNIFICANCE  \
2  39792745           Efficacy          yes
4  39792745           Efficacy          yes
5  30700996           Efficacy          yes
8  26075078           Efficacy          yes
9  22328925           Efficacy          yes

                                               NOTES  \
2  "KCNQ1 gene polymorphisms also significantly a…
4  "CYP2C9 gene polymorphisms also significantly …
5  "After treatment with exenatide, only CT/TT in…
8  A multivariate logistic model showed that the …
9  This genotype is associated with sustained vir…

                                            SENTENCE ALLELE_PharmGKB  \
2  Genotype GG is associated with decreased respo…              GG
4  Genotype TT is associated with decreased respo…              TT
5  Genotypes CT + TT is associated with increased…         CT + TT
8  Genotype TT is associated with increased respo…              TT
9  Genotype TT is associated with increased respo…              TT

  ALLELE_23andme
2              T
4              C
5              C
8              T
9              T
```

[404]:
```
# define the exact column order to keep in the exported table

cols = ["dbSNP_ID", "GENE_SYMBOL", "DRUG_NAME", "NOTES", "SENTENCE",␣
 ↪"ALLELE_PharmGKB", "ALLELE_23andme"]

# save as tab-separated .tsv file in the project folder, the address can be␣
 ↪changed
eff_sig[cols].to_csv(
    "/mnt/c/Users/yingz/Documents/BIOINF0575/Project_option_1/
 ↪23andme_PharmGKB_map.tsv",
    sep = "\t", # use tab characters as the field delimiter so the file is a TSV
    index = False # omit the DataFrame index because it is not a data column
)
eff_sig.head()
```

[404]:
```
    dbSNP_ID GENE_SYMBOL                                         DRUG_NAME  \
2   rs163184       KCNQ1                                        sitagliptin
```

```
4  rs1799853      CYP2C9                                         sitagliptin
5  rs7903146      TCF7L2                                           exenatide
8  rs8099917       IFNL3  peginterferon alfa-2a, peginterferon alfa-2b, …
9  rs8099917       IFNL3                    peginterferon alfa-2b, ribavirin

       PMID PHENOTYPE_CATEGORY SIGNIFICANCE  \
2  39792745           Efficacy          yes
4  39792745           Efficacy          yes
5  30700996           Efficacy          yes
8  26075078           Efficacy          yes
9  22328925           Efficacy          yes

                                              NOTES  \
2  "KCNQ1 gene polymorphisms also significantly a…
4  "CYP2C9 gene polymorphisms also significantly …
5  "After treatment with exenatide, only CT/TT in…
8  A multivariate logistic model showed that the …
9  This genotype is associated with sustained vir…

                                           SENTENCE ALLELE_PharmGKB  \
2  Genotype GG is associated with decreased respo…              GG
4  Genotype TT is associated with decreased respo…              TT
5  Genotypes CT + TT is associated with increased…         CT + TT
8  Genotype TT is associated with increased respo…              TT
9  Genotype TT is associated with increased respo…              TT

  ALLELE_23andme
2              T
4              C
5              C
8              T
9              T
```

[405]:
```
# build a narrow table of the three needed columns and remove rows with any␣
↪missing value

tmp = eff_sig[["GENE_SYMBOL", "DRUG_NAME", "dbSNP_ID"]].dropna()

# create a per-(gene, drug) summary that lists unique dbSNP IDs joined by␣
↪semicolons

summary = (
    tmp.groupby(["GENE_SYMBOL", "DRUG_NAME"])["dbSNP_ID"] # group rows by gene␣
↪and drug, focusing on the rsID column
       .apply(lambda s: ";".join(s.astype("string").unique())) # convert IDs to␣
↪strings, take unique values, and join them with ";"
       .reset_index() # turn the groupby index back into regular columns
```

```
        .rename(columns = {"dbSNP_ID": "dbSNP_IDs"}) # rename the aggregated␣
 ↪column to reflect that it contains multiple IDs
)

print(list(summary.columns))  # should be ['GENE_SYMBOL', 'DRUG_NAME',␣
 ↪'dbSNP_IDs']
print("Contains commas:", summary["dbSNP_IDs"].str.contains(",", na = False).
 ↪any())  # should be False
print("Spaces around semicolon:", summary["dbSNP_IDs"].str.contains(r"\s;|;\s",␣
 ↪regex = True, na = False).any()) # should be False



# save as tab-separated .tsv to the linux folder
summary.to_csv(
    "/mnt/c/Users/yingz/Documents/BIOINF0575/Project_option_1/
 ↪23andme_PharmGKB_summary.tsv",
    sep = "\t",
    index = False
)

summary.head()
```

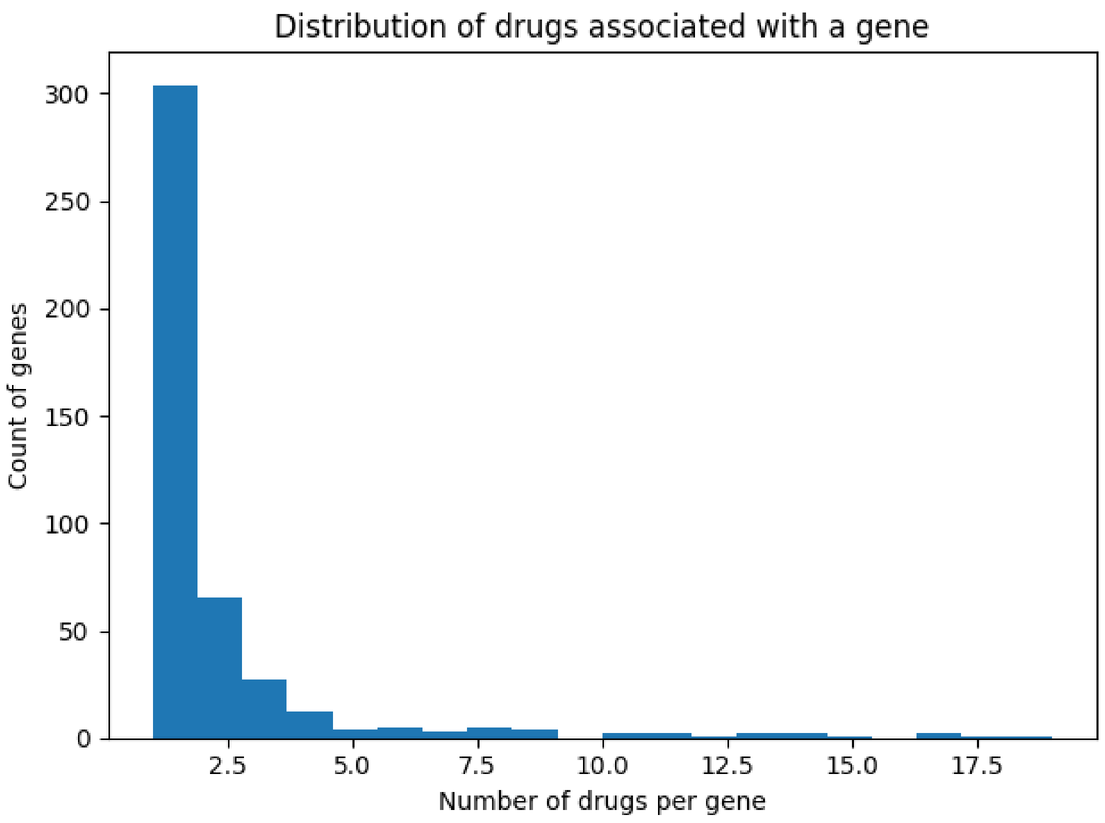
```
['GENE_SYMBOL', 'DRUG_NAME', 'dbSNP_IDs']
Contains commas: False
Spaces around semicolon: False
```

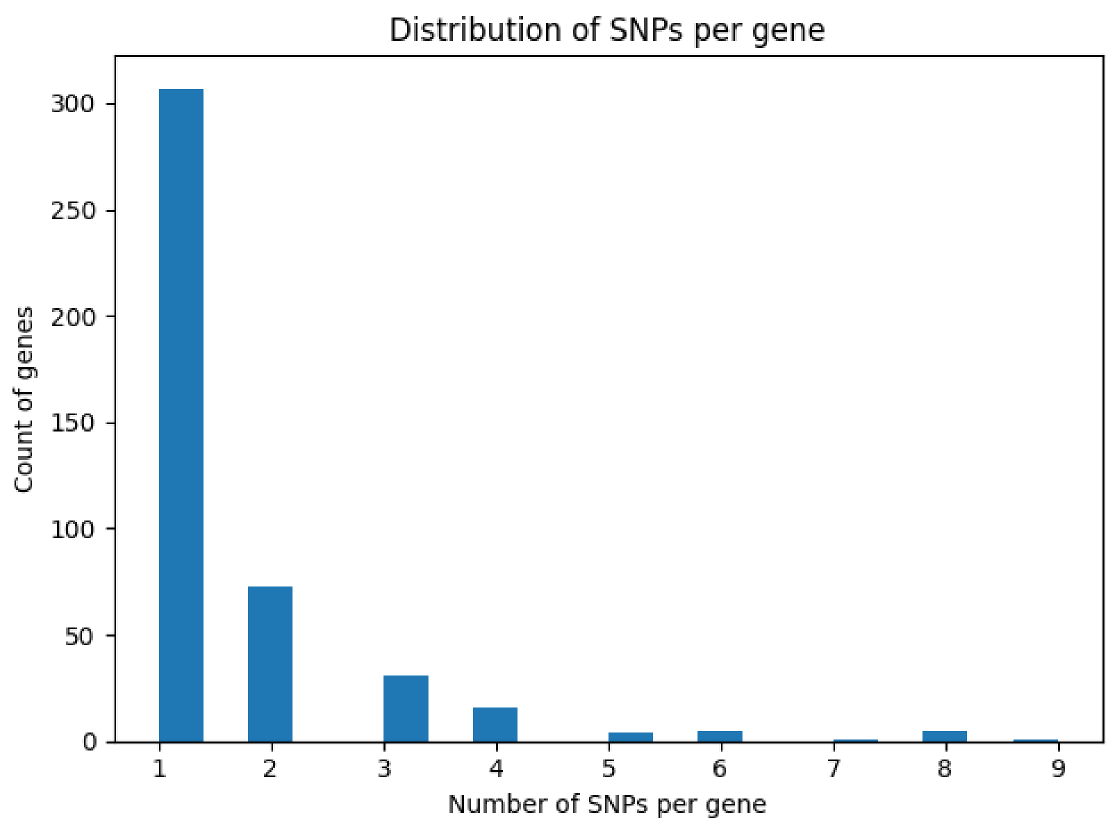
[405]:

```
  GENE_SYMBOL                                DRUG_NAME            dbSNP_IDs
0       ABCA1  atorvastatin, rosuvastatin, simvastatin            rs2230806
1       ABCA1                              fenofibrate  rs2230806;rs2230808
2       ABCB1                          antidepressants            rs1128503
3       ABCB1                           antipsychotics            rs1128503
4       ABCB1                            carbamazepine            rs1128503
```

[406]:

```
# compute, the number of distinct drugs linked to that gene for each gene

drugs_per_gene = eff_sig.groupby("GENE_SYMBOL")["DRUG_NAME"].nunique()

# compute, the number of distinct SNP rsIDs linked to that gene for each gene

snps_per_gene = eff_sig.groupby("GENE_SYMBOL")["dbSNP_ID"].nunique()

plt.figure() # start a new figure so the following plot is isolated
plt.hist(drugs_per_gene.values, bins=20) # draw a histogram of the per-gene␣
 ↪drug counts using 20 bins
plt.xlabel("Number of drugs per gene")
plt.ylabel("Count of genes")
plt.title("Distribution of drugs associated with a gene")
plt.tight_layout() # adjust margins so that labels and title fit within the␣
 ↪figure area

plt.figure()
plt.hist(snps_per_gene.values, bins=20) # draw a histogram of the per-gene SNP␣
 ↪counts using 20 bins
plt.xlabel("Number of SNPs per gene")
plt.ylabel("Count of genes")
plt.title("Distribution of SNPs per gene")
plt.tight_layout() # adjust margins so that labels and title fit within the␣
 ↪figure area
```





Feature 1: "Which drugs in this person's data are potentially affected?"

- **Biological question:** Given the person's 23andMe alleles, which drugs have at least one significant efficacy-associated SNP where the reported PharmGKB allele/genotype matches the person's allele?
- **What the feature does:** 1. Parses the PharmGKB allele text to pull out genotype tokens (e.g., TT, CT, G, etc.). 2. Flags a match when the person's ALLELE_23andme is consistent with any PharmGKB token for that SNP. 3. Aggregates by drug to show how many matched SNPs the person carries that may influence efficacy.
- **How the question be answered:** The analysis produces a prioritized list of drugs with counts of matched, significant efficacy variants observed in the person's data, which therapies may be pharmacogenomically relevant for this individual.

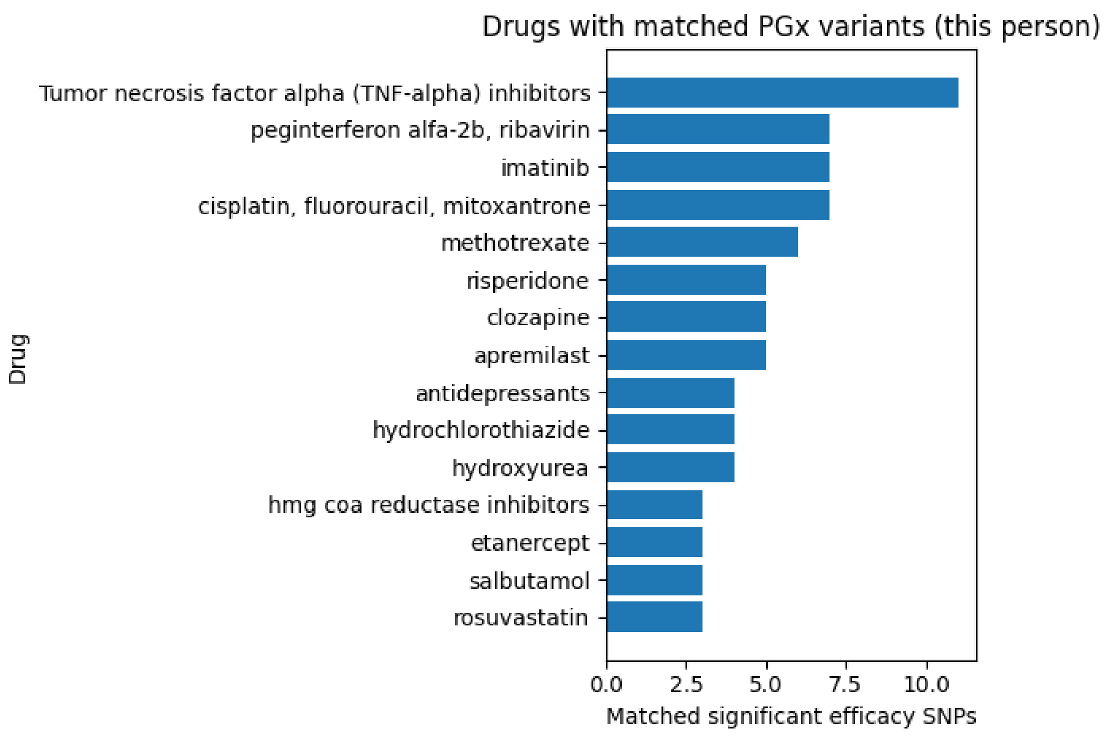
[407]:
```
# 1. # extract all alleles from PharmGKB entries as individual base tokens (A,␣
↪C, G, T, TT)

tokens = (
    eff_sig["ALLELE_PharmGKB"]
    .astype("string").str.upper().str.findall(r"[ACGT]{1,2}") # convert to␣
↪uppercase and extract 1-2 letter nucleotide strings
)

# 2. standardize the allele strings from 23andMe for comparison

allele_person = (
    eff_sig["ALLELE_23andme"]
    .astype("string").str.upper().str.strip().fillna("") # convert to␣
 ↪uppercase, remove spaces, and replace missing values with empty strings
)

# 3. determine if each individual’s allele matches any of the PharmGKB alleles␣
 ↪for that SNP

has_match = (
    pd.DataFrame({"p": allele_person, "xs": tokens})
      .apply(lambda r: (r["p"] in r["xs"]) if isinstance(r["xs"], list) else␣
 ↪False, axis=1)
) # return True only when a valid list exists and contains the allele

# 4. add a new Boolean column MATCH to indicate whether a match was found

eff_sig_matched = eff_sig.assign(MATCH=has_match)

# 5. calculate how many significant SNPs and matched SNPs each drug has

drug_hits = (
    eff_sig_matched.groupby("DRUG_NAME") # group by drug
      .agg( # compute summary statistics within each group
          n_matched = ("MATCH", lambda s: s.astype(int).sum()), # count how␣
 ↪many SNPs matched the person’s allele
          n_sig_snps = ("dbSNP_ID", "nunique") # count the number of unique␣
 ↪significant SNPs linked to the drug
      )
      .reset_index() # restore DRUG_NAME as a column
      .sort_values(["n_matched", "n_sig_snps"], ascending = False) # order␣
 ↪drugs by number of matches and significance
)

drug_hits.head(20)
```

[407]:

```
                                           DRUG_NAME  n_matched  n_sig_snps
27   Tumor necrosis factor alpha (TNF-alpha) inhibi…         11          31
316                     peginterferon alfa-2b, ribavirin           7          21
247                                         imatinib           7          17
139                cisplatin, fluorouracil, mitoxantrone           7          16
283                                     methotrexate           6          27
344                                      risperidone           5          26
```

```
156                             clozapine         5        16
66                             apremilast         5        10
59                         antidepressants         4        17
241                     hydrochlorothiazide         4        15
242                             hydroxyurea         4         6
239              hmg coa reductase inhibitors         3        20
205                              etanercept         3        13
351                              salbutamol         3        13
350                            rosuvastatin         3        12
270                                 lithium         3        11
301                              olanzapine         3        11
284                          methylphenidate         3        10
88                               benazepril         3         9
188                    docetaxel, thalidomide         3         8
```

[408]:
```python
topN = drug_hits.query("n_matched > 0").head(15) # keep only drugs with at␣
↪least one matched SNP and take the top 15 rows

plt.figure()
plt.barh(topN["DRUG_NAME"], topN["n_matched"]) # draw a horizontal bar chart␣
↪with drug names on the y-axis and matched counts on the x-axis
plt.xlabel("Matched significant efficacy SNPs")
plt.ylabel("Drug")
plt.title("Drugs with matched PGx variants (this person)")
plt.gca().invert_yaxis() # invert the y-axis so the largest values appear at␣
↪the top
plt.tight_layout()
```



Feature 2: "Which genes carry the strongest PGx signal in this person?"

Biological question: Across all genes, where does this person carry the most significant efficacy-associated variants?

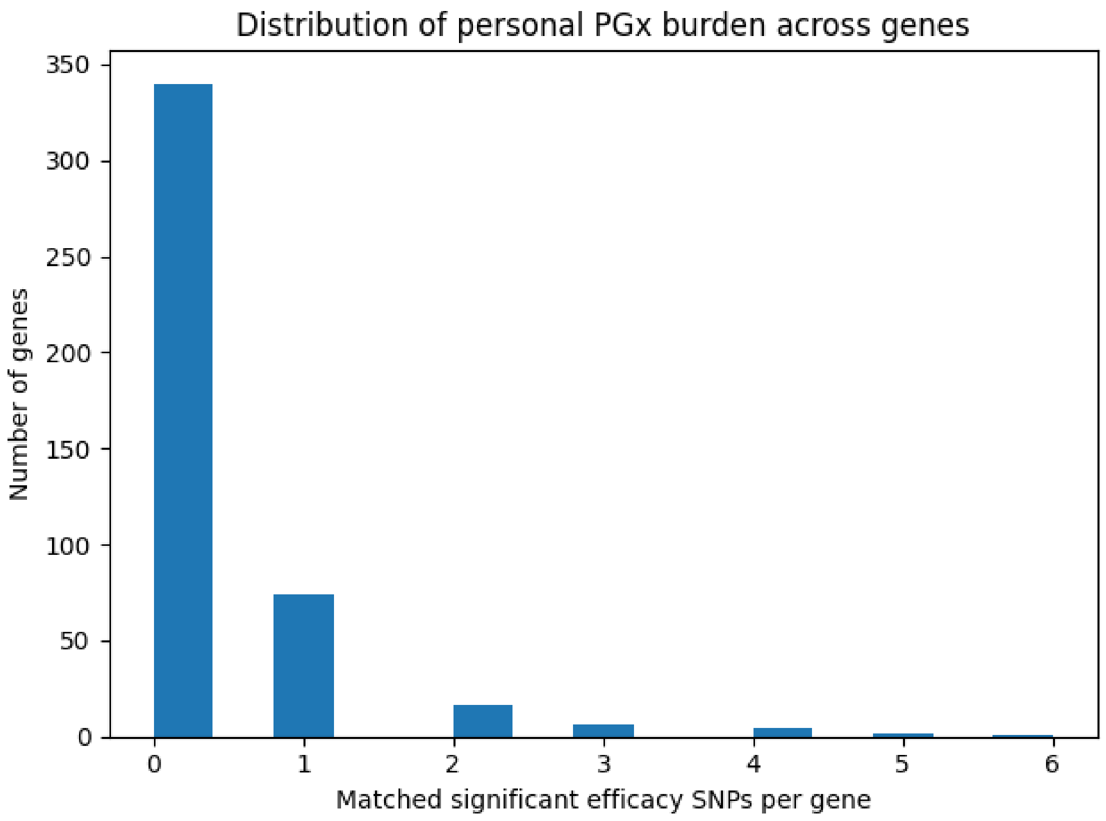
What the feature does: 1. Uses the same match flag to count, per gene, how many matched significant efficacy SNPs are present. 2. Also reports how many distinct drugs are implicated by those variants for each gene.

How it answers the question: It points to genes with the highest personal pharmacogenomic burden (more matched significant variants and drugs). These are natural focal points for interpretation or follow-up.

[409]:
```
# create a per-gene summary of matched SNPs and distinct drugs

gene_signal = (
    eff_sig_matched.groupby("GENE_SYMBOL") # group by gene symbol
                   .agg(n_matched_snps = ("MATCH", "sum"), # count matched SNPs␣
 ↪for each gene by summing the Boolean MATCH column
                        n_drugs = ("DRUG_NAME", "nunique")) # count the number␣
 ↪of distinct drugs associated with each gene
                   .reset_index() # restore GENE_SYMBOL as a regular column
                   .sort_values(["n_matched_snps","n_drugs"], ascending =␣
 ↪False) # order genes by matched SNP count, then by drug count, in descending␣
 ↪order
)

gene_signal.head(20)
```

[409]:
```
       GENE_SYMBOL  n_matched_snps  n_drugs
410            TNF               6       10
115           COMT               5       19
140         CYP3A4               5       17
1            ABCB1               4       18
224          IFNL3               4       12
361        SLC19A1               4        4
53          BCL11A               4        1
225   IFNL3, IFNL4               3       15
6            ABCG2               3        9
142         CYP3A5               3        7
207          GSTP1               3        7
119          CRHR2               3        3
379        SLCO1C1               3        3
22           ADRB2               2       17
57            BDNF               2       13
276          MTHFR               2       11
172         FCGR2A               2        8
24             AGT               2        7
437          XRCC1               2        6
20          ADRA2A               2        4
```

[410]:
```
plt.figure()
plt.hist(gene_signal["n_matched_snps"].values, bins=15) # draw a histogram of␣
 ↪the number of matched SNPs per gene using 15 bins
plt.xlabel("Matched significant efficacy SNPs per gene")
plt.ylabel("Number of genes")
plt.title("Distribution of personal PGx burden across genes")
plt.tight_layout()
```



Feature 3: "Dimensionality Reduction and Predictive Profiling of Personal PGx Patterns."

Biological question: Do this person's matched pharmacogenomic variants cluster into interpretable patterns across genes and drugs, and can these structures help distinguish high- versus low-burden genes?

What the feature does:

- Builds a gene–drug matrix from the person's efficacy-significant matches (rows = genes, columns = drugs, values = matched variant counts). After filtering and variance selection, the working table is 103 genes × 30 drugs (class distribution for the label used below: low burden = 79, high burden = 24).
- Computes a drug–drug correlation heatmap to visualize co-occurring PGx signals. The map is mostly sparse but shows localized patches (e.g., TNF- inhibitors, interferons, antidepressants) indicating shared response patterns.
- Applies PCA (after standardization) to reduce the matrix into interpretable axes. The first two PCs explain 7.3% and 6.8% of variance, respectively, revealing a modest latent structure with a few outlier genes contributing disproportionately to variation.
- Trains a logistic regression classifier to separate high- vs low-burden genes. On a held-out test set, overall accuracy 0.81. Class-wise performance: Low burden (0) shows precision 0.83, recall 0.96, F1 0.89 (n=25), while High burden (1) shows precision 0.50, recall 0.17, F1 0.25

(n=6). The confusion matrix confirms strong performance on the majority class and reduced sensitivity for high-burden genes due to class imbalance.

How it answers the question: The pipeline uncovers system-level patterns in the individual's PGx profile: the correlation map highlights drug clusters with similar variant signatures; PCA summarizes these signals into personal PGx axes; and the classifier tests whether these axes carry predictive information about variant burden. In this person, structure is present but subtle (low explained variance, clustered points with a few outliers), and prediction favors the prevalent low-burden class. Together, the results provide a concise global summary of the person's PGx architecture and clarify where additional data (e.g., more balanced classes or richer features) would improve detection of high-burden signals.

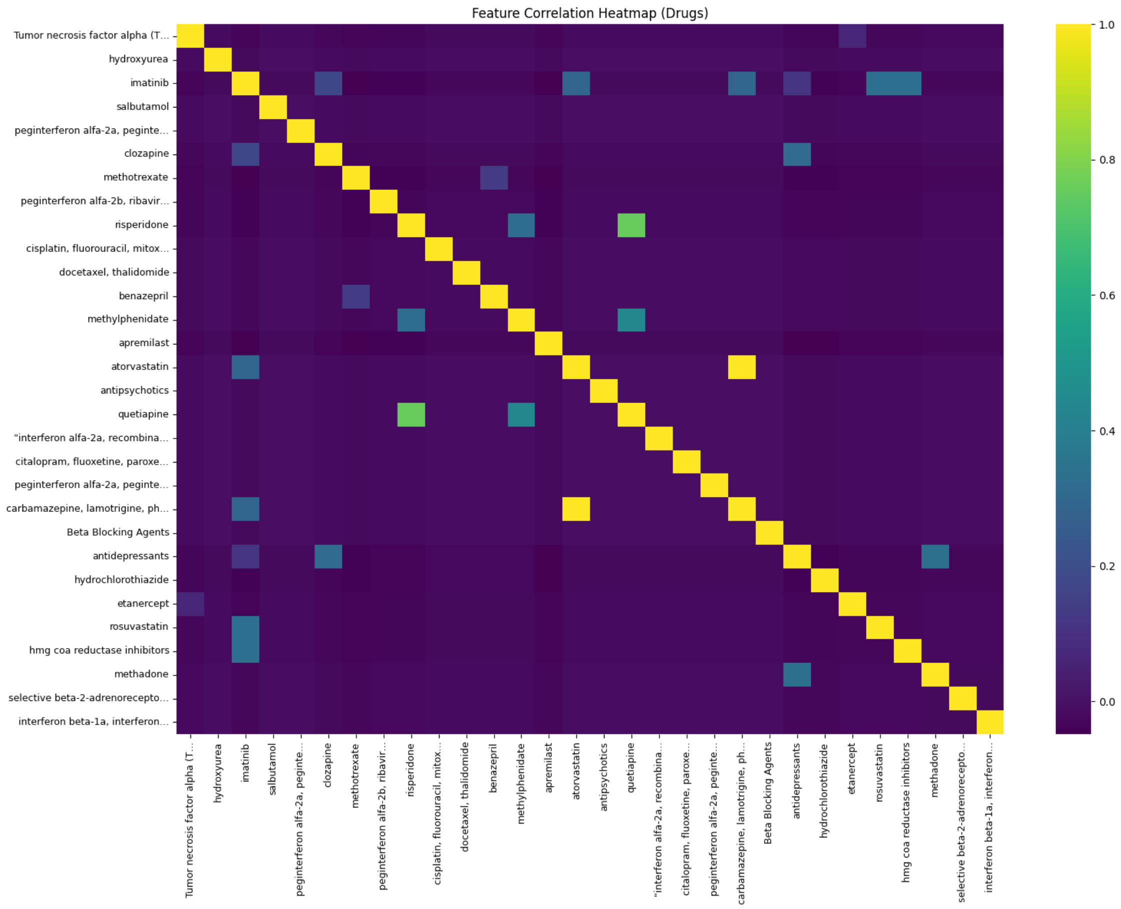
[411]:
```
# build a gene-drug table from matched rows

matrix = (eff_sig_matched # personal efficacy-significant rows
          .groupby(["GENE_SYMBOL", "DRUG_NAME"])["MATCH"] # group by gene and␣
 ↪drug, and then take MATCH column
          .sum() # sum matches within each gene-drug pair
          .unstack(fill_value = 0) # pivot to wide: rows = genes, cols = drugs,␣
 ↪fill missing with 0
         )

# filter matrix to keep only informative rows/cols

x = matrix.loc[matrix.sum(axis = 1) > 0, # rows: genes with at least one␣
 ↪nonzero count
               matrix.sum(axis = 0) > 0 # cols: drugs with at least one nonzero␣
 ↪count
              ].copy() # avoid chained-assignment warnings

if x.shape[1] > 30: # keep at most 30 most-variable drugs
    top_cols = x.var(axis=0).sort_values(ascending=False).head(30).index  #␣
 ↪pick by variance
    x = x.loc[:, top_cols] # subset columns to those top-variance drugs

burden = x.sum(axis = 1) # per-gene total matched count across all drugs
y = (burden > burden.median()).astype(int) # binary label: 1 = high burden␣
 ↪(above median), 0 = low

print(f"Shape of PGx matrix: genes = {x.shape[0]}, drugs(features) = {x.
 ↪shape[1]}")
print("Class distribution (0 = low burden, 1 = high burden):")
print(y.value_counts()) # counts of low vs high burden genes
```

```
Shape of PGx matrix: genes = 103, drugs(features) = 30
Class distribution (0 = low burden, 1 = high burden):
0    79
1    24
```

```
Name: count, dtype: int64
```

The dataset is imbalanced, with roughly 3 times more low-burden (79) than high-burden (24) genes, which is common and suggests that only a subset of genes show strong multi-drug Pharmacogenomics (PGx) signals.

```
[ ]: corr = x.corr() # compute feature-feature (drug-drug) correlation matrix

     def short(s, n = 30): # shorten very long labels for display
         return (s[:n] + '…') if len(s) > n else s

     corr_disp = corr.copy()
     corr_disp.columns = [short(c) for c in corr.columns]  # shorten x-axis labels␣
      ↪(columns)
     corr_disp.index = [short(r) for r in corr.index] # shorten y-axis labels (rows)

     plt.figure(figsize=(16, 12)) # larger size
     ax = sns.heatmap(corr_disp, cmap = 'viridis')
     plt.title('Feature Correlation Heatmap (Drugs)')
     ax.tick_params(axis = 'x', rotation = 90, labelsize = 9) # rotate x labels
     ax.tick_params(axis = 'y', labelsize = 9) # smaller y labels
     plt.tight_layout()
     plt.show()
```

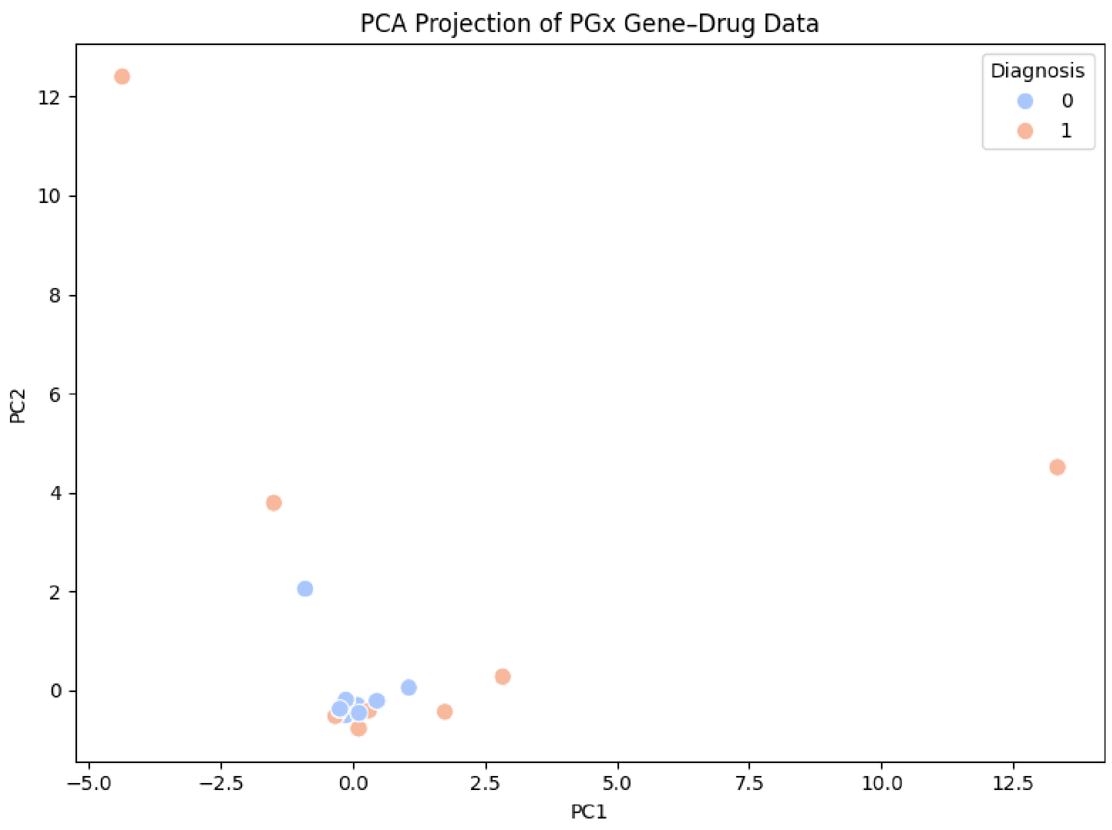


The heatmap illustrates pairwise correlations among 30 drug features derived from the pharmacogenomic (PGx) gene–drug matrix. Each cell represents the Pearson correlation between two drugs, reflecting the extent to which they share efficacy-related variant patterns across genes.

Overall, most drug pairs show near-zero correlations, indicating that pharmacogenomic effects are largely drug-specific. Only a few localized clusters exhibit moderate positive correlations, suggesting partially overlapping PGx mechanisms—likely through shared metabolic enzymes (e.g., CYP family genes) or common biological pathways. These results confirm that the pharmacogenomic landscape is heterogeneous across drugs. The observed correlation structure provides the basis for dimensionality reduction through PCA, which summarizes shared PGx variance into interpretable components for subsequent classification analysis.

```
[ ]: scaler = StandardScaler() # create a StandardScaler to normalize each feature
     X_scaled = scaler.fit_transform(x) # scale drug-gene matrix so every feature
      ↪has mean = 0, sd = 1

     pca = PCA(n_components = 2, random_state = 1997) # set up PCA to reduce data to
      ↪2 principal components
```

```
principal_components = pca.fit_transform(X_scaled) # compute PCs from the␣
↪standardized matrix
pca_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'],␣
↪index=x.index) # make a dataframe of PC1 and PC2 scores
pca_df['Diagnosis'] = y.values # add the binary outcome (high vs low burden)␣
↪for plotting

plt.figure(figsize = (8, 6))
sns.scatterplot(
    x = 'PC1',
    y = 'PC2',
    hue = 'Diagnosis', # color points by Diagnosis (0 = low, 1 = high)
    data = pca_df,
    s = 80, # point size in the scatterplot
    palette = 'coolwarm' # color map
)
plt.title('PCA Projection of PGx Gene-Drug Data')
plt.tight_layout()
plt.show()

# report proportion of total variance captured by PC1, PC2

print('Explained variance by first two PCs:', np.round(pca.
↪explained_variance_ratio_, 3))
```



```
Explained variance by first two PCs: [0.073 0.068]
```

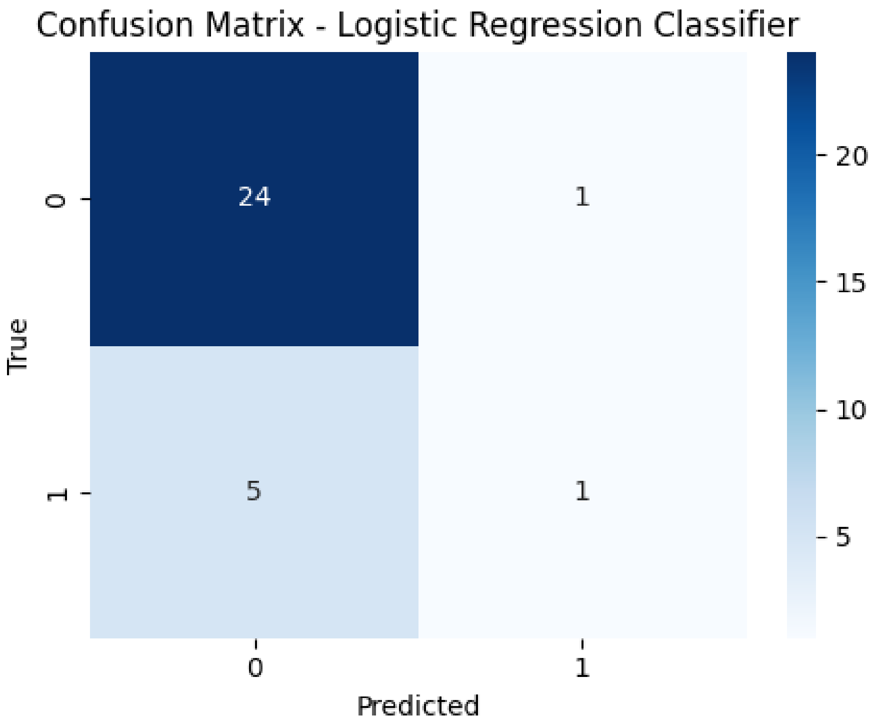
This plot displays the first two principal components (PC1 and PC2) derived from the standardized pharmacogenomic (PGx) gene–drug matrix. Each point represents an individual sample, color-coded by PGx burden status (0 = low burden, 1 = high burden).

The first two components capture the dominant variance patterns across drug-specific PGx features. Most samples cluster tightly near the origin, indicating that the majority share similar overall PGx profiles. A few scattered points lie farther along the PC1 or PC2 axes, suggesting the presence of outlier individuals whose pharmacogenomic signatures are driven by distinct gene–drug interactions.

While no clear linear separation between low- and high-burden classes is observed in the two-dimensional space, the dispersion along PC1 and PC2 reflects meaningful heterogeneity in drug-related genetic variability. These components provide a compact, noise-reduced representation for downstream classification analyses, such as logistic regression.

```
[ ]: # split features/labels into train/test with 30% test set
     X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.
      ↪3, random_state = 1997)
     logit = LogisticRegression( # logistic regression classifier
         solver = 'lbfgs', # LBFGS optimizer for medium dataset
         max_iter = 1000
)

logit.fit(X_train, y_train)
y_pred = logit.predict(X_test) # predict class labels on the test set

print("\nClassification report:")
print(classification_report(y_test, y_pred)) # precision/recall/F1/support by␣
 ↪class

plt.figure(figsize = (5,4))
sns.heatmap(
    confusion_matrix(y_test, y_pred), # matrix that rows = true, cols =␣
 ↪predicted
    annot = True, # show counts in each cell
    fmt = 'd', # integer formatting for annotations
    cmap = 'Blues'
)
plt.title('Confusion Matrix - Logistic Regression Classifier')
plt.xlabel('Predicted'); plt.ylabel('True')
plt.tight_layout()
plt.show()
```

```
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.96      0.89        25
           1       0.50      0.17      0.25         6

    accuracy                           0.81        31
   macro avg       0.66      0.56      0.57        31
weighted avg       0.76      0.81      0.77        31
```



The logistic regression model achieved an overall accuracy of 0.81 in distinguishing between low and high PGx burden individuals based on the top drug–gene features. The confusion matrix indicates that most low-burden samples (class 0) were correctly classified (24 out of 25), while the model struggled to identify high-burden cases (class 1), with only one correctly predicted and five misclassified as low-burden.

Precision for the low-burden group (0.83) and high recall (0.96) suggest strong specificity, meaning the model reliably identifies individuals without extensive pharmacogenomic interactions. In contrast, the high-burden group shows limited sensitivity (recall = 0.17, F1 = 0.25), reflecting the class imbalance in the dataset (approximately 3:1 ratio).

Overall, the classifier captures dominant PGx patterns characterizing the low-burden population but lacks sufficient signal to robustly detect high-burden individuals. These findings highlight the need for either feature enrichment (e.g., inclusion of additional drug–gene pairs) or model regularization adjustments to improve sensitivity in identifying clinically relevant PGx outliers.