```
import os

for dirname, _, filenames in os.walk("/kaggle/input"):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/gse66695-series-matrix/GSE66695_series_matrix.txt
```

```
file_path = "/kaggle/input/gse66695-series-matrix/GSE66695_series_matrix.txt"
```

```
import pandas as pd
from io import StringIO

# Read the file
with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
    lines = f.readlines()

# Find table boundaries
begin = end = None
for i, line in enumerate(lines):
    s = line.strip()
    if s == "!series_matrix_table_begin":
        begin = i + 1
    elif s == "!series_matrix_table_end":
        end = i
        break

if begin is None or end is None:
    raise RuntimeError("Could not find !series_matrix_table_begin/end")

print("Table rows:", end - begin)

# Extract and load table
table_text = "".join(lines[begin:end])
df = pd.read_csv(StringIO(table_text), sep="\t", low_memory=False)

print("Loaded table shape:", df.shape)
print("First 10 columns:", list(df.columns)[:10])
df.head(3)
```

```
Table rows: 485578
Loaded table shape: (485577, 121)
First 10 columns: ['ID_REF', 'GSM1629194', 'GSM1629195', 'GSM1629196', 'GSM1629197', 'GSM1629198', 'GSM1629199', 'GSM1629200', '
```

	ID_REF	GSM1629194	GSM1629195	GSM1629196	GSM1629197	GSM1629198	GSM1629199	GSM1629200	GSM1629201	GSM1629202	...	G
0	cg00000029	0.334061	0.178954	0.172730	0.183075	0.072717	0.195457	0.287615	0.339056	0.195401	...	
1	cg00000108	0.935733	0.937377	0.929570	0.924546	0.922173	0.880320	0.909696	0.929620	0.903362	...	
2	cg00000109	0.874611	0.738671	0.741349	0.856688	0.871578	0.686862	0.730466	0.850274	0.712961	...	

3 rows × 121 columns

```
import re
import numpy as np

# Set probe ID as index
df["ID_REF"] = df["ID_REF"].astype(str)
df = df.set_index("ID_REF")

# Select GSM sample columns
gsm_cols = [c for c in df.columns if re.match(r"^GSM\d+", str(c))]
print("Number of samples:", len(gsm_cols))

X_full = df[gsm_cols].apply(pd.to_numeric, errors="coerce")

print("X_full shape (CpGs x samples):", X_full.shape)
print("Overall missing fraction:", X_full.isna().mean().mean())
```

```
Number of samples: 120
X_full shape (CpGs x samples): (485577, 120)
Overall missing fraction: 0.002712631913510456
```

```
# Extract sample metadata from the series matrix header (no internet needed)

sample_meta = {}

for line in lines:
    if line.startswith("!Sample_"):
        key, values = line.strip().split("\t", 1)
        values = values.split("\t")
        sample_meta[key] = values

# Inspect available metadata fields
print("Available Sample metadata keys:")
for k in sample_meta.keys():
    print(k)
```

```
# 1) Confirm we have the file contents in `lines`
print("Total lines in file:", len(lines))
print("First 5 lines:")
for x in lines[:5]:
    print(x.strip())

# 2) Find *any* lines that mention "Sample" in the header (before the table begins)
table_begin_idx = None
for i, line in enumerate(lines):
    if line.strip() == "!series_matrix_table_begin":
        table_begin_idx = i
        break

print("\nTable begins at line:", table_begin_idx)

sample_lines = []
for line in lines[:table_begin_idx]:
    if "Sample" in line or "sample" in line:
        sample_lines.append(line.strip())

print("\nFound", len(sample_lines), "header lines containing 'Sample'. Showing up to 30:\n")
for s in sample_lines[:30]:
    print(s[:200])  # truncate so it doesn't spam
```

```
Total lines in file: 485642
First 5 lines:
!Series_title   "Genome-wide Scan for Methylation Profiles in Breast Cancer"
!Series_geo_accession   "GSE66695"
!Series_status  "Public on Mar 11 2015"
!Series_submission_date "Mar 09 2015"
!Series_last_update_date        "Mar 22 2019"

Table begins at line: 62

Found 36 header lines containing 'Sample'. Showing up to 30:

!Series_summary "Genome wide DNA methylation profiling of normal and breast cancer samples. The Illumina Infinium 450k Human DNA
!Series_overall_design  "Bisulphite converted DNA from the 120 samples were hybridised to the Illumina Infinium 450k Human Methy
!Series_sample_id       "GSM1629194 GSM1629195 GSM1629196 GSM1629197 GSM1629198 GSM1629199 GSM1629200 GSM1629201 GSM1629202 GSM1
!Series_sample_taxid    "9606"
!Sample_title   "01-208-T"      "01-254-N"      "01-212-N"      "01-254-T"      "01-212-T"      "01-265-N"      "01-237-N"
!Sample_geo_accession   "GSM1629194"    "GSM1629195"    "GSM1629196"    "GSM1629197"    "GSM1629198"    "GSM1629199"    "GSM1629
!Sample_status  "Public on Mar 11 2015" "Public on Mar 11 2015" "Public on Mar 11 2015" "Public on Mar 11 2015" "Public on Mar 1
!Sample_submission_date "Mar 09 2015"   "Mar 09 2015"   "Mar 09 2015"   "Mar 09 2015"   "Mar 09 2015"   "Mar 09 2015"   "Mar 09
!Sample_last_update_date        "Mar 11 2015"   "Mar 11 2015"   "Mar 11 2015"   "Mar 11 2015"   "Mar 11 2015"   "Mar 11 2015"
!Sample_type    "genomic"       "genomic"       "genomic"       "genomic"       "genomic"       "genomic"       "genomic"
!Sample_channel_count   "1"     "1"     "1"     "1"     "1"     "1"     "1"     "1"     "1"     "1"     "1"     "1"
!Sample_source_name_ch1 "Tumor" "Normal"        "Normal"        "Tumor" "Tumor" "Normal"        "Normal"        "Tumor" "Normal"
!Sample_organism_ch1    "Homo sapiens"  "Homo sapiens"  "Homo sapiens"  "Homo sapiens"  "Homo sapiens"  "Homo sapiens"  "Homo sa
!Sample_characteristics_ch1     "gender: Female"        "gender: Female"        "gender: Female"        "gender: Female"
!Sample_characteristics_ch1     "sample type: ER Negative"      "sample type: Normal"   "sample type: Normal"   "sample type: ER
!Sample_characteristics_ch1     "disease state: Tumor"  "disease state: Normal" "disease state: Normal" "disease state: Tumor"
!Sample_molecule_ch1    "genomic DNA"   "genomic DNA"   "genomic DNA"   "genomic DNA"   "genomic DNA"   "genomic DNA"   "genomic
!Sample_extract_protocol_ch1    "genomic DNA was extracted and purified from breast cancer samples using Qiagen DNeasy Kit accor
!Sample_label_ch1       "Cy5 and Cy3"   "Cy5 and Cy3"   "Cy5 and Cy3"   "Cy5 and Cy3"   "Cy5 and Cy3"   "Cy5 and Cy3"   "Cy5 and
!Sample_label_protocol_ch1      "Standard Illumina Protocol"    "Standard Illumina Protocol"    "Standard Illumina Protocol"
!Sample_taxid_ch1       "9606"  "9606"  "9606"  "9606"  "9606"  "9606"  "9606"  "9606"  "9606"  "9606"  "9606"  "9606"
!Sample_hyb_protocol    "bisulphite converted DNA was amplified, fragmented and hybridised to Illumina Infinium Human Methylatio
!Sample_scan_protocol   "Arrays were imaged using BeadArray Reader using standard recommended Illumina scanner setting" "Arrays
!Sample_data_processing "GenomeStudio V2011.1"  "GenomeStudio V2011.1"  "GenomeStudio V2011.1"  "GenomeStudio V2011.1"  "GenomeS
!Sample_data_processing "Unmethylated and methylated signal intensities"        "Unmethylated and methylated signal intensities"
!Sample_platform_id     "GPL13534"      "GPL13534"      "GPL13534"      "GPL13534"      "GPL13534"      "GPL13534"      "GPL1353
!Sample_contact_name    "Maria,,Worsham"        "Maria,,Worsham"        "Maria,,Worsham"        "Maria,,Worsham"        "Maria,,
!Sample_contact_department      "Otolaryngology"        "Otolaryngology"        "Otolaryngology"        "Otolaryngology"
!Sample_contact_institute       "Henry Ford Health System"      "Henry Ford Health System"      "Henry Ford Health System"
```

```
!Sample_contact_address "1 Ford Place"  "1 Ford Place"  "1 Ford Place"  "1 Ford Place"  "1 Ford Place"  "1 Ford Place"  "1 Ford
```

```
# ---- Extract sample IDs and labels from header ----

gsm_ids = None
groups = None

for line in lines:
    if line.startswith("!Sample_geo_accession"):
        gsm_ids = line.strip().split("\t")[1:]
    if line.startswith("!Sample_source_name_ch1"):
        groups = line.strip().split("\t")[1:]

# Sanity checks
print("Found GSM IDs:", len(gsm_ids))
print("Found group labels:", len(groups))

print("First 10 GSM IDs:", gsm_ids[:10])
print("First 10 group labels:", groups[:10])
```

```
Found GSM IDs: 120
Found group labels: 120
First 10 GSM IDs: ['"GSM1629194"', '"GSM1629195"', '"GSM1629196"', '"GSM1629197"', '"GSM1629198"', '"GSM1629199"', '"GSM1629200"
First 10 group labels: ['"Tumor"', '"Normal"', '"Normal"', '"Tumor"', '"Tumor"', '"Normal"', '"Normal"', '"Tumor"', '"Normal"',
```

```
# Build phenotype dataframe
pheno = pd.DataFrame({
    "GSM": gsm_ids,
    "Group": groups
})

# Keep only Tumor / Normal
pheno = pheno[pheno["Group"].isin(["Tumor", "Normal"])].reset_index(drop=True)

print("Label counts:")
print(pheno["Group"].value_counts())

pheno.head()
```

```
Label counts:
Series([], Name: count, dtype: int64)
```

	GSM	Group

```
# Rebuild pheno cleanly (handles quotes + whitespace)
pheno = pd.DataFrame({"GSM": gsm_ids, "Group": groups})

# Clean the strings: remove quotes and extra spaces
pheno["GSM"] = pheno["GSM"].astype(str).str.strip().str.strip('"')
pheno["Group"] = pheno["Group"].astype(str).str.strip().str.strip('"')

# Quick debug: see what values actually exist
print("Unique Group values (first 10):", pheno["Group"].unique()[:10])

# Keep only Tumor/Normal
pheno = pheno[pheno["Group"].isin(["Tumor", "Normal"])].reset_index(drop=True)

print("\nLabel counts:")
print(pheno["Group"].value_counts())
pheno.head()
```

```
Unique Group values (first 10): ['Tumor' 'Normal']

Label counts:
Group
Tumor     80
Normal    40
Name: count, dtype: int64
```

	GSM	Group
0	GSM1629194	Tumor
1	GSM1629195	Normal
2	GSM1629196	Normal
3	GSM1629197	Tumor
4	GSM1629198	Tumor

```
X = X_full[pheno["GSM"]]
y = (pheno["Group"] == "Tumor").astype(int)

print("Final X shape:", X.shape)
print("Tumor proportion:", float(y.mean()))
```

```
Final X shape: (485577, 120)
Tumor proportion: 0.6666666666666666
```

```
# Drop CpGs with >10% missing values
keep = X.isna().mean(axis=1) <= 0.10
X = X.loc[keep]

# Median imputation per CpG
X = X.apply(lambda r: r.fillna(r.median()), axis=1)

print("After cleanup X shape:", X.shape)
```

```
After cleanup X shape: (485200, 120)
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_classif

# Transpose for sklearn (samples x CpGs)
X_s = X.T

X_train, X_test, y_train, y_test = train_test_split(
    X_s, y, test_size=0.25, stratify=y, random_state=42
)

# Remove near-constant CpGs
varf = VarianceThreshold(threshold=1e-5)
X_train_v = varf.fit_transform(X_train)
X_test_v  = varf.transform(X_test)

# Select top CpGs
K = min(2000, X_train_v.shape[1])
skb = SelectKBest(f_classif, k=K)

X_train_sel = skb.fit_transform(X_train_v, y_train)
X_test_sel  = skb.transform(X_test_v)

print("Selected CpGs:", X_train_sel.shape[1])
```

```
Selected CpGs: 2000
```

```
clf = Pipeline([
    ("scaler", StandardScaler()),
    ("lr", LogisticRegression(
        penalty="l1",
        solver="saga",
        C=0.2,                   # stronger regularization than default 1.0
        max_iter=20000,
        tol=1e-3,
        class_weight="balanced",
```

```
        random_state=42,
        n_jobs=-1
    ))
])

clf.fit(X_train_sel, y_train)
print("Model trained.")
```

```
Model trained.
```

```
lr = clf.named_steps["lr"]
print("Iterations used:", lr.n_iter_)
print("Non-zero coefficients:", int((lr.coef_ != 0).sum()))
```

```
Iterations used: [636]
Non-zero coefficients: 87
```

```
from sklearn.metrics import roc_auc_score, average_precision_score, confusion_matrix, classification_report

proba = clf.predict_proba(X_test_sel)[:, 1]

print("ROC AUC:", roc_auc_score(y_test, proba))
print("PR AUC:", average_precision_score(y_test, proba))

pred = (proba >= 0.5).astype(int)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, pred))
print("\nClassification Report:\n", classification_report(y_test, pred, digits=3))
```

```
ROC AUC: 1.0
PR AUC: 1.0000000000000002

Confusion Matrix:
 [[10  0]
 [ 0 20]]

Classification Report:
              precision    recall  f1-score   support

           0      1.000     1.000     1.000        10
           1      1.000     1.000     1.000        20

    accuracy                          1.000        30
   macro avg      1.000     1.000     1.000        30
weighted avg      1.000     1.000     1.000        30
```

```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_classif
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([
    ("var", VarianceThreshold(1e-5)),
    ("skb", SelectKBest(f_classif, k=2000)),
    ("scaler", StandardScaler()),
    ("lr", LogisticRegression(
        penalty="l1",
        solver="saga",
        C=0.2,
        max_iter=20000,
        tol=1e-3,
        class_weight="balanced",
        random_state=42
    ))
])

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

cv_auc = cross_val_score(pipe, X_s, y, cv=cv, scoring="roc_auc")
cv_pr  = cross_val_score(pipe, X_s, y, cv=cv, scoring="average_precision")

print("Nested CV ROC AUC:", cv_auc)
print("Mean ± SD:", cv_auc.mean(), "±", cv_auc.std())
```

```
print("\nNested CV PR AUC:", cv_pr)
print("Mean ± SD:", cv_pr.mean(), "±", cv_pr.std())
```

```
Nested CV ROC AUC: [1.        1.        0.9765625 1.        1.       ]
Mean ± SD: 0.9953125 ± 0.009375

Nested CV PR AUC: [1.         1.         0.98825061 1.         1.        ]
Mean ± SD: 0.9976501225490196 ± 0.004699754901960773
```

```
import os

for dirname, _, filenames in os.walk("/kaggle/input/gpl13534-humanmethylation450"):
    for f in filenames:
        print(os.path.join(dirname, f))
```

```
/kaggle/input/gpl13534-humanmethylation450/GPL13534_HumanMethylation450_15017482_v.1.1.csv
```

```
import pandas as pd

annot_path = "/kaggle/input/gpl13534-humanmethylation450/GPL13534_HumanMethylation450_15017482_v.1.1.csv"
```

```
import os

for dirname, _, filenames in os.walk("/kaggle/input/gpl13534-humanmethylation450"):
    print("DIR:", dirname)
    for f in filenames:
        print("  ", f)
```

```
DIR: /kaggle/input/gpl13534-humanmethylation450
   GPL13534_HumanMethylation450_15017482_v.1.1.csv
```

```
import pandas as pd

annot = pd.read_csv(annot_path)
print("Annotation shape:", annot.shape)
annot.head(3)
```

```
---------------------------------------------------------------------------
ParserError                               Traceback (most recent call last)
/tmp/ipykernel_55/3965123890.py in <cell line: 0>()
      1 import pandas as pd
      2
----> 3 annot = pd.read_csv(annot_path)
      4 print("Annotation shape:", annot.shape)
      5 annot.head(3)

 3 frames
/usr/local/lib/python3.12/dist-packages/pandas/io/parsers/c_parser_wrapper.py in read(self, nrows)
    232         try:
    233             if self.low_memory:
--> 234                 chunks = self._reader.read_low_memory(nrows)
    235                 # destructive to chunks
    236                 data = _concatenate_chunks(chunks)

parsers.pyx in pandas._libs.parsers.TextReader.read_low_memory()

parsers.pyx in pandas._libs.parsers.TextReader._read_rows()

parsers.pyx in pandas._libs.parsers.TextReader._tokenize_rows()

parsers.pyx in pandas._libs.parsers.TextReader._check_tokenize_status()

parsers.pyx in pandas._libs.parsers.raise_parser_error()

ParserError: Error tokenizing data. C error: Expected 27 fields in line 8, saw 33
```

```
annot_path = "/kaggle/input/gpl13534-humanmethylation450/GPL13534_HumanMethylation450_15017482_v.1.1.csv"
```

```
with open(annot_path, "r", encoding="utf-8", errors="ignore") as f:
    for i in range(15):
        line = f.readline()
        print(f"{i+1:02d}:", line.rstrip()[:200])
```

```
01: Illumina, Inc.,,,,,,,,,,,,,,,,,,,,,,,,
02: [Heading],,,,,,,,,,,,,,,,,,,,,,,,
03: Descriptor File Name,BS0010894-AQP_content.bpm,,,,,,,,,,,,,,,,,,,,,,,,
04: Assay Format,Infinium 2,,,,,,,,,,,,,,,,,,,,,,,,
05: Date Manufactured,6/11/2008,,,,,,,,,,,,,,,,,,,,,,,,
06: Loci Count ,485553,,,,,,,,,,,,,,,,,,,,,,,,
07: [Assay],,,,,,,,,,,,,,,,,,,,,,,,
08: IlmnID,Name,AddressA_ID,AlleleA_ProbeSeq,AddressB_ID,AlleleB_ProbeSeq,Infinium_Design_Type,Next_Base,Color_Channel,Forward_S
09: cg00035864,cg00035864,31729416,AAAACACTAACAATCTTATCCACATAAACCCTTAAATTTATCTCAAATTC,,,II,,,AATCCAAAGATGATGGAGGAGTGCCCGCTCATGAT
10: cg00050873,cg00050873,32735311,ACAAAAAAACAACACACAACTATAATAATTTTTAAAATAAATAAACCCCA,31717405,ACGAAAAAACAACGCACAACTATAATAATTTTT
11: cg00061679,cg00061679,28780415,AAAACATTAAAAAACTAATTCACTACTATTTAATTACTTTATTTTCCATC,,,II,,,TCAACAAATGAGAGACATTGAAGAACTAATTCACT
12: cg00063477,cg00063477,16712347,TATTCTTCCACACAAAATACTAAACRTATATTTACAAAAATACTTCCATC,,,II,,,CTCCTGTACTTGTTCATTAAATAATGATTCCTTGG
13: cg00121626,cg00121626,19779393,AAAACTAATAAAAATAACTTACAAACCAAATACTATACCCTACAACTCTC,,,II,,,AGGTGAATGAAGAGACTAATGGGAGTGGCTTGCAA
14: cg00212031,cg00212031,29674443,CCCAATTAACCACAAAAACTAAACAAATTATACAATCAAAAAAACATACA,38703326,CCCAATTAACCGCAAAAACTAAACAAATTATAC
15: cg00213748,cg00213748,30703409,TTTTAACACCTAACACCATTTTAACAATAAAAATTCTACAAAAAAAAACA,36767301,TTTTAACGCCTAACACCGTTTTAACGATAAAAA
```

```
import pandas as pd

annot = pd.read_csv(
    annot_path,
    sep="\t",
    engine="python",
    encoding_errors="ignore"
)

print("Loaded with TAB. Shape:", annot.shape)
annot.head(3)
```

```
Loaded with TAB. Shape: (486435, 1)
```

	Illumina, Inc.,,,,,,,,,,,,,,,,,,,,,,,,
0	[Heading],,,,,,,,,,,,,,,,,,,,,,,,
1	Descriptor File Name,BS0010894-AQP_content.bpm...
2	Assay Format,Infinium 2,,,,,,,,,,,,,,,,,,,,,,,,

```
annot_path = "/kaggle/input/gpl13534-humanmethylation450/GPL13534_HumanMethylation450_15017482_v.1.1.csv"

with open(annot_path, "r", encoding="utf-8", errors="ignore") as f:
    for i, line in enumerate(f):
        if "IlmnID" in line and "UCSC_RefGene_Name" in line:
            print("Header found at line number (0-based):", i)
            print("Header preview:", line[:200])
            header_line = i
            break
```

```
Header found at line number (0-based): 7
Header preview: IlmnID,Name,AddressA_ID,AlleleA_ProbeSeq,AddressB_ID,AlleleB_ProbeSeq,Infinium_Design_Type,Next_Base,Color_Chann
```

```
import pandas as pd

annot = pd.read_csv(
    annot_path,
    sep=",",
    engine="python",
    skiprows=header_line
)

print("Annotation shape:", annot.shape)
print("First columns:", list(annot.columns)[:15])
annot.head(3)
```

```
Annotation shape: (486428, 33)
First columns: ['IlmnID', 'Name', 'AddressA_ID', 'AlleleA_ProbeSeq', 'AddressB_ID', 'AlleleB_ProbeSeq', 'Infinium_Design_Type',
```

	IlmnID	Name	AddressA_ID	AlleleA_ProbeSeq	AddressB_ID	
0	cg00035864	cg00035864	31729416	AAAACACTAACAATCTTATCCACATAAACCCTTAAATTTATCTCAA...	NaN	
1	cg00050873	cg00050873	32735311	ACAAAAAAACAACACACAACTATAATAATTTTTAAAATAAATAAAC...	31717405	ACGAAAAAACAACC
2	cg00061679	cg00061679	28780415	AAAACATTAAAAAACTAATTCACTACTATTTAATTACTTTATTTTC...	NaN	

3 rows × 33 columns

```
annot_sub = annot[[
    "IlmnID",
    "UCSC_RefGene_Name",
    "UCSC_RefGene_Group"
]].copy()

annot_sub = annot_sub.rename(columns={"IlmnID": "CpG"})

annot_sub.head(3)
```

	CpG	UCSC_RefGene_Name	UCSC_RefGene_Group
0	cg00035864	TTTY18	TSS1500
1	cg00050873	TSPY4;FAM197Y2	Body;TSS1500
2	cg00061679	DAZ1;DAZ4;DAZ4	Body;Body;Body

```
top20_annot = top20.merge(annot_sub, on="CpG", how="left")

top20_annot
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_55/291599245.py in <cell line: 0>()
----> 1 top20_annot = top20.merge(annot_sub, on="CpG", how="left")
      2
      3 top20_annot

NameError: name 'top20' is not defined
```

```
import numpy as np
import pandas as pd

# 1) Recover CpG names after feature selection
probe_names = X_s.columns.to_numpy()           # all CpGs
probes_after_var = probe_names[varf.get_support()]
final_probes = probes_after_var[skb.get_support()]

# 2) Get model coefficients
coef = clf.named_steps["lr"].coef_.ravel()

# 3) Rank CpGs by absolute coefficient
top_idx = np.argsort(np.abs(coef))[::-1]

top20 = pd.DataFrame({
    "CpG": final_probes[top_idx],
    "Coefficient": coef[top_idx],
    "AbsCoeff": np.abs(coef[top_idx])
}).head(20).reset_index(drop=True)

top20
```

	CpG	Coefficient	AbsCoeff
0	cg25137968	0.146332	0.146332
1	cg11436222	0.108071	0.108071
2	cg17239057	0.107967	0.107967
3	cg13068653	0.103593	0.103593
4	cg13193196	0.101761	0.101761
5	cg09684846	0.083109	0.083109
6	cg25683810	0.082379	0.082379
7	cg04320956	0.079678	0.079678
8	cg21503345	0.077985	0.077985
9	cg12305431	0.077141	0.077141
10	cg20192747	0.075405	0.075405
11	cg11609668	0.070063	0.070063
12	cg01507342	-0.066476	0.066476
13	cg22671717	0.065312	0.065312
14	cg18641463	0.061756	0.061756
15	cg03130910	-0.059055	0.059055
16	cg16845701	0.057336	0.057336
17	cg16708174	0.056524	0.056524
18	cg10441424	-0.051610	0.051610
19	cg15427886	0.051339	0.051339

```
top20_annot = top20.merge(annot_sub, on="CpG", how="left")
top20_annot
```

	CpG	Coefficient	AbsCoeff	UCSC_RefGene_Name	UCSC_RefGene_Group
0	cg25137968	0.146332	0.146332	HIST2H2BA	TSS1500
1	cg11436222	0.108071	0.108071	HIST1H3E	1stExon
2	cg17239057	0.107967	0.107967	NaN	NaN
3	cg13068653	0.103593	0.103593	TAGLN;TAGLN	Body;Body
4	cg13193196	0.101761	0.101761	IMMP2L	Body
5	cg09684846	0.083109	0.083109	FAM18A	TSS200
6	cg25683810	0.082379	0.082379	NaN	NaN
7	cg04320956	0.079678	0.079678	HAS3;HAS3	Body;Body
8	cg21503345	0.077985	0.077985	NaN	NaN
9	cg12305431	0.077141	0.077141	HOXA3;HOXA3	5'UTR;5'UTR
10	cg20192747	0.075405	0.075405	NaN	NaN
11	cg11609668	0.070063	0.070063	TAGLN;TAGLN	3'UTR;3'UTR
12	cg01507342	-0.066476	0.066476	PITPNC1;PITPNC1	Body;Body
13	cg22671717	0.065312	0.065312	NaN	NaN
14	cg18641463	0.061756	0.061756	MAD1L1;MAD1L1;MAD1L1	Body;Body;Body
15	cg03130910	-0.059055	0.059055	NaN	NaN
16	cg16845701	0.057336	0.057336	NaN	NaN
17	cg16708174	0.056524	0.056524	RARRES1;RARRES1	Body;Body
18	cg10441424	-0.051610	0.051610	NaN	NaN
19	cg15427886	0.051339	0.051339	NaN	NaN

```
# Promoter vs gene body
def region_type(group):
    if pd.isna(group):
        return "Intergenic"
    if any(x in group for x in ["TSS200", "TSS1500", "5'UTR", "1stExon"]):
        return "Promoter"
    return "Gene body / Other"

top20_annot["RegionType"] = top20_annot["UCSC_RefGene_Group"].apply(region_type)

# Hyper vs hypo methylation
top20_annot["Tumor_Methylation"] = top20_annot["Coefficient"].apply(
    lambda x: "Hypermethylated in Tumor" if x > 0 else "Hypomethylated in Tumor"
)

final_table = top20_annot[[
    "CpG",
    "UCSC_RefGene_Name",
    "RegionType",
    "Tumor_Methylation",
    "Coefficient"
]]

final_table
```

	CpG	UCSC_RefGene_Name	RegionType	Tumor_Methylation	Coefficient
0	cg25137968	HIST2H2BA	Promoter	Hypermethylated in Tumor	0.146332
1	cg11436222	HIST1H3E	Promoter	Hypermethylated in Tumor	0.108071
2	cg17239057	NaN	Intergenic	Hypermethylated in Tumor	0.107967
3	cg13068653	TAGLN;TAGLN	Gene body / Other	Hypermethylated in Tumor	0.103593
4	cg13193196	IMMP2L	Gene body / Other	Hypermethylated in Tumor	0.101761
5	cg09684846	FAM18A	Promoter	Hypermethylated in Tumor	0.083109
6	cg25683810	NaN	Intergenic	Hypermethylated in Tumor	0.082379
7	cg04320956	HAS3;HAS3	Gene body / Other	Hypermethylated in Tumor	0.079678
8	cg21503345	NaN	Intergenic	Hypermethylated in Tumor	0.077985
9	cg12305431	HOXA3;HOXA3	Promoter	Hypermethylated in Tumor	0.077141
10	cg20192747	NaN	Intergenic	Hypermethylated in Tumor	0.075405
11	cg11609668	TAGLN;TAGLN	Gene body / Other	Hypermethylated in Tumor	0.070063
12	cg01507342	PITPNC1;PITPNC1	Gene body / Other	Hypomethylated in Tumor	-0.066476
13	cg22671717	NaN	Intergenic	Hypermethylated in Tumor	0.065312
14	cg18641463	MAD1L1;MAD1L1;MAD1L1	Gene body / Other	Hypermethylated in Tumor	0.061756
15	cg03130910	NaN	Intergenic	Hypomethylated in Tumor	-0.059055
16	cg16845701	NaN	Intergenic	Hypermethylated in Tumor	0.057336
17	cg16708174	RARRES1;RARRES1	Gene body / Other	Hypermethylated in Tumor	0.056524
18	cg10441424	NaN	Intergenic	Hypomethylated in Tumor	-0.051610
19	cg15427886	NaN	Intergenic	Hypermethylated in Tumor	0.051339

```
# Curated list of well-known cancer / cancer-related genes
# (conservative, literature-supported)
known_cancer_genes = {
    "HOXA3",
    "TAGLN",
    "RARRES1",
    "MAD1L1",
    "IMMP2L",
    "HAS3",
    "HIST1H3E",
    "HIST2H2BA",
    "PITPNC1"
}

def is_cancer_related(gene_str):
```

```
    if pd.isna(gene_str):
        return "Unknown / Intergenic"
    genes = gene_str.split(";")
    return "Yes" if any(g in known_cancer_genes for g in genes) else "Possibly / Unknown"

final_table = final_table.copy()
final_table.loc[:, "Cancer_Related_Gene"] = final_table["UCSC_RefGene_Name"].apply(is_cancer_related)

final_table
```

	CpG	UCSC_RefGene_Name	RegionType	Tumor_Methylation	Coefficient	Cancer_Related_Gene
0	cg25137968	HIST2H2BA	Promoter	Hypermethylated in Tumor	0.146332	Yes
1	cg11436222	HIST1H3E	Promoter	Hypermethylated in Tumor	0.108071	Yes
2	cg17239057	NaN	Intergenic	Hypermethylated in Tumor	0.107967	Unknown / Intergenic
3	cg13068653	TAGLN;TAGLN	Gene body / Other	Hypermethylated in Tumor	0.103593	Yes
4	cg13193196	IMMP2L	Gene body / Other	Hypermethylated in Tumor	0.101761	Yes
5	cg09684846	FAM18A	Promoter	Hypermethylated in Tumor	0.083109	Possibly / Unknown
6	cg25683810	NaN	Intergenic	Hypermethylated in Tumor	0.082379	Unknown / Intergenic
7	cg04320956	HAS3;HAS3	Gene body / Other	Hypermethylated in Tumor	0.079678	Yes
8	cg21503345	NaN	Intergenic	Hypermethylated in Tumor	0.077985	Unknown / Intergenic
9	cg12305431	HOXA3;HOXA3	Promoter	Hypermethylated in Tumor	0.077141	Yes
10	cg20192747	NaN	Intergenic	Hypermethylated in Tumor	0.075405	Unknown / Intergenic
11	cg11609668	TAGLN;TAGLN	Gene body / Other	Hypermethylated in Tumor	0.070063	Yes
12	cg01507342	PITPNC1;PITPNC1	Gene body / Other	Hypomethylated in Tumor	-0.066476	Yes
13	cg22671717	NaN	Intergenic	Hypermethylated in Tumor	0.065312	Unknown / Intergenic
14	cg18641463	MAD1L1;MAD1L1;MAD1L1	Gene body / Other	Hypermethylated in Tumor	0.061756	Yes
15	cg03130910	NaN	Intergenic	Hypomethylated in Tumor	-0.059055	Unknown / Intergenic
16	cg16845701	NaN	Intergenic	Hypermethylated in Tumor	0.057336	Unknown / Intergenic
17	cg16708174	RARRES1;RARRES1	Gene body / Other	Hypermethylated in Tumor	0.056524	Yes
18	cg10441424	NaN	Intergenic	Hypomethylated in Tumor	-0.051610	Unknown / Intergenic
19	cg15427886	NaN	Intergenic	Hypermethylated in Tumor	0.051339	Unknown / Intergenic

```
final_table["Cancer_Related_Gene"].value_counts()
```

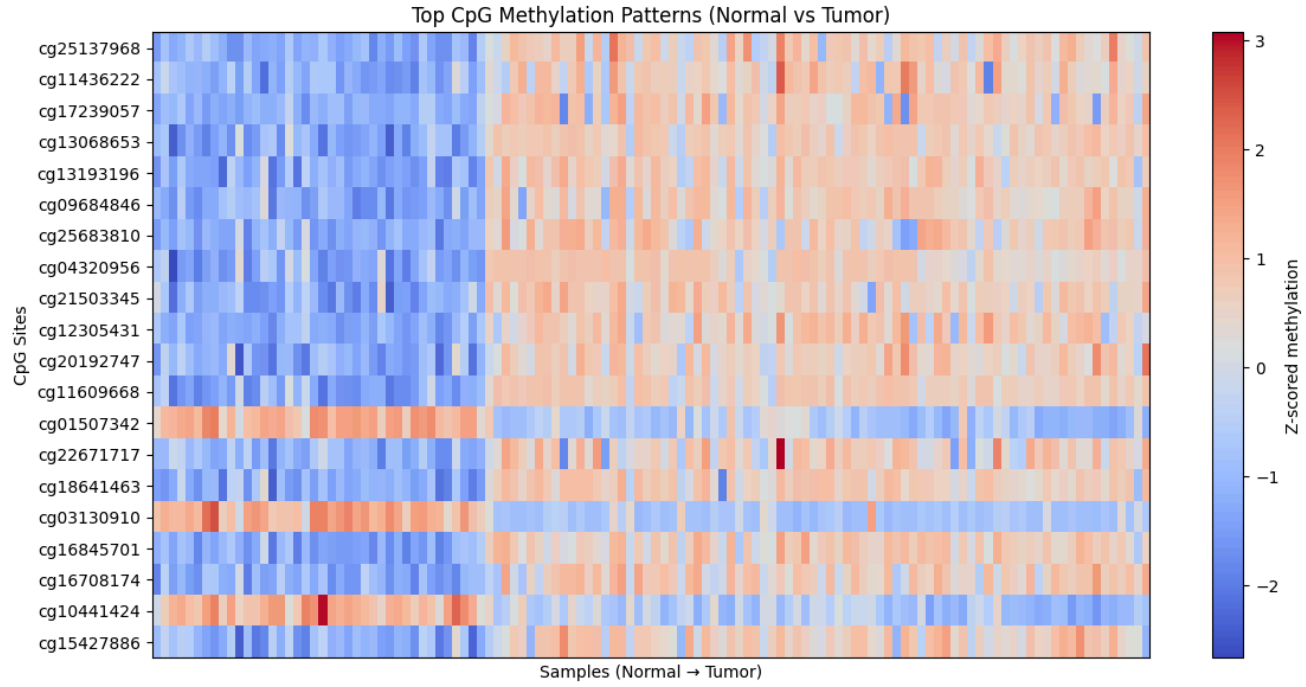
```
Cancer_Related_Gene
Yes                    10
Unknown / Intergenic    9
Possibly / Unknown      1
Name: count, dtype: int64
```

```
# Extract beta values for top CpGs
top_cpg_ids = final_table["CpG"].tolist()

X_top = X.loc[top_cpg_ids]  # CpGs x samples

# Reorder samples: Normal first, Tumor second
sample_order = pheno.sort_values("Group")["GSM"]
X_top = X_top[sample_order]

# Z-score CpGs for visualization
X_top_z = (X_top - X_top.mean(axis=1).values.reshape(-1,1)) / X_top.std(axis=1).values.reshape(-1,1)

X_top_z.shape
```

```
(20, 120)
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
```

```
plt.imshow(X_top_z, aspect="auto", cmap="coolwarm")
plt.colorbar(label="Z-scored methylation")

# Labels
plt.yticks(range(len(top_cpg_ids)), top_cpg_ids)
plt.xticks([])

plt.title("Top CpG Methylation Patterns (Normal vs Tumor)")
plt.ylabel("CpG Sites")
plt.xlabel("Samples (Normal → Tumor)")

plt.tight_layout()
plt.show()
```




```
# X_s should be Samples x CpGs (120 x ~485k)
print("X_s shape:", X_s.shape)
print("pheno shape:", pheno.shape)

# y must be aligned to X_s rows (same order)
y_aligned = pheno.set_index("GSM").loc[X_s.index, "Group"].eq("Tumor").astype(int)

print("y_aligned length:", len(y_aligned))
print("Class counts:", y_aligned.value_counts().to_dict())
```

```
X_s shape: (120, 485200)
pheno shape: (120, 2)
y_aligned length: 120
Class counts: {1: 80, 0: 40}
```

```
from sklearn.model_selection import StratifiedKFold, cross_val_score
import numpy as np

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

n_permutations = 25
perm_scores = []

for i in range(n_permutations):
    y_perm = np.random.permutation(y_aligned.values)
    score = cross_val_score(
        pipe,          # IMPORTANT: use the full pipeline with feature selection inside
        X_s,
        y_perm
```

```
            y_perm,
            cv=cv,
            scoring="roc_auc",
            n_jobs=-1
        ).mean()
        perm_scores.append(score)

print("Permutation ROC AUC (mean ± SD):", float(np.mean(perm_scores)), "±", float(np.std(perm_scores)))
```