

Start coding or [upload](#) with AI.

Double-click (or enter) to edit

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

The Final project is a study of diabetic patients Dataset.

Dr:Hager Saleh

Student:Jihad Abdullah Alharibi,253002789

Dataset description

```
import warnings
warnings.filterwarnings("ignore")
# =====
# Core libraries
# =====
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# =====
# Load data - data handling
# =====
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

# =====
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, classification_report
# =====
# Imbalanced learn: sampling methods
# =====
from imblearn.over_sampling import RandomOverSampler, SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.combine import SMOTETomek

df = pd.read_csv('content/diabetes.csv')

df.dtypes

#
# Pregnancies int64
# Glucose int64
# BloodPressure int64
# SkinThickness int64
# Insulin int64
# BMI float64
# DiabetesPedigreeFunction float64
# Age int64
# Outcome int64
dtype: object

df.head()

#
# Pregnancies int64
# Glucose int64
# BloodPressure int64
# SkinThickness int64
# Insulin int64
# BMI float64
# DiabetesPedigreeFunction float64
# Age int64
# Outcome int64
dtype: object

df['Glucose'].value_counts()

#
# count
# Glucose
# 99 17
# 100 17
# 101 14
# 102 14
# 103 14
# 104 14
# 105 1
# 106 1
# 107 1
# 108 1
# 109 1
# 110 1
# 111 1
# 112 1
# 113 1
# 114 1
# 115 1
# 116 1
# 117 1
# 118 1
# 119 1
# 120 1
# 121 1
# 122 1
# 123 1
# 124 1
# 125 1
# 126 1
# 127 1
# 128 1
# 129 1
# 130 1
# 131 1
# 132 1
# 133 1
# 134 1
# 135 1
# 136 1
# 137 1
# 138 1
# 139 1
# 140 1
# 141 1
# 142 1
# 143 1
# 144 1
# 145 1
# 146 1
# 147 1
# 148 1
# 149 1
# 150 1
# 151 1
# 152 1
# 153 1
# 154 1
# 155 1
# 156 1
# 157 1
# 158 1
# 159 1
# 160 1
# 161 1
# 162 1
# 163 1
# 164 1
# 165 1
# 166 1
# 167 1
# 168 1
# 169 1
# 170 1
# 171 1
# 172 1
# 173 1
# 174 1
# 175 1
# 176 1
# 177 1
# 178 1
# 179 1
# 180 1
# 181 1
# 182 1
# 183 1
# 184 1
# 185 1
# 186 1
# 187 1
# 188 1
# 189 1
# 190 1
# 191 1
# 192 1
# 193 1
# 194 1
# 195 1
# 196 1
# 197 1
# 198 1
# 199 1
# 200 1
# 201 1
# 202 1
# 203 1
# 204 1
# 205 1
# 206 1
# 207 1
# 208 1
# 209 1
# 210 1
# 211 1
# 212 1
# 213 1
# 214 1
# 215 1
# 216 1
# 217 1
# 218 1
# 219 1
# 220 1
# 221 1
# 222 1
# 223 1
# 224 1
# 225 1
# 226 1
# 227 1
# 228 1
# 229 1
# 230 1
# 231 1
# 232 1
# 233 1
# 234 1
# 235 1
# 236 1
# 237 1
# 238 1
# 239 1
# 240 1
# 241 1
# 242 1
# 243 1
# 244 1
# 245 1
# 246 1
# 247 1
# 248 1
# 249 1
# 250 1
# 251 1
# 252 1
# 253 1
# 254 1
# 255 1
# 256 1
# 257 1
# 258 1
# 259 1
# 260 1
# 261 1
# 262 1
# 263 1
# 264 1
# 265 1
# 266 1
# 267 1
# 268 1
# 269 1
# 270 1
# 271 1
# 272 1
# 273 1
# 274 1
# 275 1
# 276 1
# 277 1
# 278 1
# 279 1
# 280 1
# 281 1
# 282 1
# 283 1
# 284 1
# 285 1
# 286 1
# 287 1
# 288 1
# 289 1
# 290 1
# 291 1
# 292 1
# 293 1
# 294 1
# 295 1
# 296 1
# 297 1
# 298 1
# 299 1
# 300 1
# 301 1
# 302 1
# 303 1
# 304 1
# 305 1
# 306 1
# 307 1
# 308 1
# 309 1
# 310 1
# 311 1
# 312 1
# 313 1
# 314 1
# 315 1
# 316 1
# 317 1
# 318 1
# 319 1
# 320 1
# 321 1
# 322 1
# 323 1
# 324 1
# 325 1
# 326 1
# 327 1
# 328 1
# 329 1
# 330 1
# 331 1
# 332 1
# 333 1
# 334 1
# 335 1
# 336 1
# 337 1
# 338 1
# 339 1
# 340 1
# 341 1
# 342 1
# 343 1
# 344 1
# 345 1
# 346 1
# 347 1
# 348 1
# 349 1
# 350 1
# 351 1
# 352 1
# 353 1
# 354 1
# 355 1
# 356 1
# 357 1
# 358 1
# 359 1
# 360 1
# 361 1
# 362 1
# 363 1
# 364 1
# 365 1
# 366 1
# 367 1
# 368 1
# 369 1
# 370 1
# 371 1
# 372 1
# 373 1
# 374 1
# 375 1
# 376 1
# 377 1
# 378 1
# 379 1
# 380 1
# 381 1
# 382 1
# 383 1
# 384 1
# 385 1
# 386 1
# 387 1
# 388 1
# 389 1
# 390 1
# 391 1
# 392 1
# 393 1
# 394 1
# 395 1
# 396 1
# 397 1
# 398 1
# 399 1
# 400 1
# 401 1
# 402 1
# 403 1
# 404 1
# 405 1
# 406 1
# 407 1
# 408 1
# 409 1
# 410 1
# 411 1
# 412 1
# 413 1
# 414 1
# 415 1
# 416 1
# 417 1
# 418 1
# 419 1
# 420 1
# 421 1
# 422 1
# 423 1
# 424 1
# 425 1
# 426 1
# 427 1
# 428 1
# 429 1
# 430 1
# 431 1
# 432 1
# 433 1
# 434 1
# 435 1
# 436 1
# 437 1
# 438 1
# 439 1
# 440 1
# 441 1
# 442 1
# 443 1
# 444 1
# 445 1
# 446 1
# 447 1
# 448 1
# 449 1
# 450 1
# 451 1
# 452 1
# 453 1
# 454 1
# 455 1
# 456 1
# 457 1
# 458 1
# 459 1
# 460 1
# 461 1
# 462 1
# 463 1
# 464 1
# 465 1
# 466 1
# 467 1
# 468 1
# 469 1
# 470 1
# 471 1
# 472 1
# 473 1
# 474 1
# 475 1
# 476 1
# 477 1
# 478 1
# 479 1
# 480 1
# 481 1
# 482 1
# 483 1
# 484 1
# 485 1
# 486 1
# 487 1
# 488 1
# 489 1
# 490 1
# 491 1
# 492 1
# 493 1
# 494 1
# 495 1
# 496 1
# 497 1
# 498 1
# 499 1
# 500 1
# 501 1
# 502 1
# 503 1
# 504 1
# 505 1
# 506 1
# 507 1
# 508 1
# 509 1
# 510 1
# 511 1
# 512 1
# 513 1
# 514 1
# 515 1
# 516 1
# 517 1
# 518 1
# 519 1
# 520 1
# 521 1
# 522 1
# 523 1
# 524 1
# 525 1
# 526 1
# 527 1
# 528 1
# 529 1
# 530 1
# 531 1
# 532 1
# 533 1
# 534 1
# 535 1
# 536 1
# 537 1
# 538 1
# 539 1
# 540 1
# 541 1
# 542 1
# 543 1
# 544 1
# 545 1
# 546 1
# 547 1
# 548 1
# 549 1
# 550 1
# 551 1
# 552 1
# 553 1
# 554 1
# 555 1
# 556 1
# 557 1
# 558 1
# 559 1
# 560 1
# 561 1
# 562 1
# 563 1
# 564 1
# 565 1
# 566 1
# 567 1
# 568 1
# 569 1
# 570 1
# 571 1
# 572 1
# 573 1
# 574 1
# 575 1
# 576 1
# 577 1
# 578 1
# 579 1
# 580 1
# 581 1
# 582 1
# 583 1
# 584 1
# 585 1
# 586 1
# 587 1
# 588 1
# 589 1
# 590 1
# 591 1
# 592 1
# 593 1
# 594 1
# 595 1
# 596 1
# 597 1
# 598 1
# 599 1
# 600 1
# 601 1
# 602 1
# 603 1
# 604 1
# 605 1
# 606 1
# 607 1
# 608 1
# 609 1
# 610 1
# 611 1
# 612 1
# 613 1
# 614 1
# 615 1
# 616 1
# 617 1
# 618 1
# 619 1
# 620 1
# 621 1
# 622 1
# 623 1
# 624 1
# 625 1
# 626 1
# 627 1
# 628 1
# 629 1
# 630 1
# 631 1
# 632 1
# 633 1
# 634 1
# 635 1
# 636 1
# 637 1
# 638 1
# 639 1
# 640 1
# 641 1
# 642 1
# 643 1
# 644 1
# 645 1
# 646 1
# 647 1
# 648 1
# 649 1
# 650 1
# 651 1
# 652 1
# 653 1
# 654 1
# 655 1
# 656 1
# 657 1
# 658 1
# 659 1
# 660 1
# 661 1
# 662 1
# 663 1
# 664 1
# 665 1
# 666 1
# 667 1
# 668 1
# 669 1
# 670 1
# 671 1
# 672 1
# 673 1
# 674 1
# 675 1
# 676 1
# 677 1
# 678 1
# 679 1
# 680 1
# 681 1
# 682 1
# 683 1
# 684 1
# 685 1
# 686 1
# 687 1
# 688 1
# 689 1
# 690 1
# 691 1
# 692 1
# 693 1
# 694 1
# 695 1
# 696 1
# 697 1
# 698 1
# 699 1
# 700 1
# 701 1
# 702 1
# 703 1
# 704 1
# 705 1
# 706 1
# 707 1
# 708 1
# 709 1
# 710 1
# 711 1
# 712 1
# 713 1
# 714 1
# 715 1
# 716 1
# 717 1
# 718 1
# 719 1
# 720 1
# 721 1
# 722 1
# 723 1
# 724 1
# 725 1
# 726 1
# 727 1
# 728 1
# 729 1
# 730 1
# 731 1
# 732 1
# 733 1
# 734 1
# 735 1
# 736 1
# 737 1
# 738 1
# 739 1
# 740 1
# 741 1
# 742 1
# 743 1
# 744 1
# 745 1
# 746 1
# 747 1
# 748 1
# 749 1
# 750 1
# 751 1
# 752 1
# 753 1
# 754 1
# 755 1
# 756 1
# 757 1
# 758 1
# 759 1
# 760 1
# 761 1
# 762 1
# 763 1
# 764 1
# 765 1
# 766 1
# 767 1
# 768 1
# 769 1
# 770 1
# 771 1
# 772 1
# 773 1
# 774 1
# 775 1
# 776 1
# 777 1
# 778 1
# 779 1
# 780 1
# 781 1
# 782 1
# 783 1
# 784 1
# 785 1
# 786 1
# 787 1
# 788 1
# 789 1
# 790 1
# 791 1
# 792 1
# 793 1
# 794 1
# 795 1
# 796 1
# 797 1
# 798 1
# 799 1
# 800 1
# 801 1
# 802 1
# 803 1
# 804 1
# 805 1
# 806 1
# 807 1
# 808 1
# 809 1
# 810 1
# 811 1
# 812 1
# 813 1
# 814 1
# 815 1
# 816 1
# 817 1
# 818 1
# 819 1
# 820 1
# 821 1
# 822 1
# 823 1
# 824 1
# 825 1
# 826 1
# 827 1
# 828 1
# 829 1
# 830 1
# 831 1
# 832 1
# 833 1
# 834 1
# 835 1
# 836 1
# 837 1
# 838 1
# 839 1
# 840 1
# 841 1
# 842 1
# 843 1
# 844 1
# 845 1
# 846 1
# 847 1
# 848 1
# 849 1
# 850 1
# 851 1
# 852 1
# 853 1
# 854 1
# 855 1
# 856 1
# 857 1
# 858 1
# 859 1
# 860 1
# 861 1
# 862 1
# 863 1
# 864 1
# 865 1
# 866 1
# 867 1
# 868 1
# 869 1
# 870 1
# 871 1
# 872 1
# 873 1
# 874 1
# 875 1
# 876 1
# 877 1
# 878 1
# 879 1
# 880 1
# 881 1
# 882 1
# 883 1
# 884 1
# 885 1
# 886 1
# 887 1
# 888 1
# 889 1
# 890 1
# 891 1
# 892 1
# 893 1
# 894 1
# 895 1
# 896 1
# 897 1
# 898 1
# 899 1
# 900 1
# 901 1
# 902 1
# 903 1
# 904 1
# 905 1
# 906 1
# 907 1
# 908 1
# 909 1
# 910 1
# 911 1
# 912 1
# 913 1
# 914 1
# 915 1
# 916 1
# 917 1
# 918 1
# 919 1
# 920 1
# 921 1
# 922 1
# 923 1
# 924 1
# 925 1
# 926 1
# 927 1
# 928 1
# 929 1
# 930 1
# 931 1
# 932 1
# 933 1
# 934 1
# 935 1
# 936 1
# 937 1
# 938 1
# 939 1
# 940 1
# 941 1
# 942 1
# 943 1
# 944 1
# 945 1
# 946 1
# 947 1
# 948 1
# 949 1
# 950 1
# 951 1
# 952 1
# 953 1
# 954 1
# 955 1
# 956 1
# 957 1
# 958 1
# 959 1
# 960 1
# 961 1
# 962 1
# 963 1
# 964 1
# 965 1
# 966 1
# 967 1
# 968 1
# 969 1
# 970 1
# 971 1
# 972 1
# 973 1
# 974 1
# 975 1
# 976 1
# 977 1
# 978 1
# 979 1
# 980 1
# 981 1
# 982 1
# 983 1
# 984 1
# 985 1
# 986 1
# 987 1
# 988 1
# 989 1
# 990 1
# 991 1
# 992 1
# 993 1
# 994 1
# 995 1
# 996 1
# 997 1
# 998 1
# 999 1
# 1000 1
# 1001 1
# 1002 1
# 1003 1
# 1004 1
# 1005 1
# 1006 1
# 1007 1
# 1008 1
# 1009 1
# 1010 1
# 1011 1
# 1012 1
# 1013 1
# 1014 1
# 1015 1
# 1016 1
# 1017 1
# 1018 1
# 1019 1
# 1020 1
# 1021 1
# 1022 1
# 1023 1
# 1024 1
# 1025 1
# 1026 1
# 1027 1
# 1028 1
# 1029 1
# 1030 1
# 1031 1
# 1032 1
# 1033 1
# 1034 1
# 1035 1
# 1036 1
# 1037 1
# 1038 1
# 1039 1
# 1040 1
# 1041 1
# 1042 1
# 1043 1
# 1044 1
# 1045 1
# 1046 1
# 1047 1
# 1048 1
# 1049 1
# 1050 1
# 1051 1
# 1052 1
# 1053 1
# 1054 1
# 1055 1
# 1056 1
# 1057 1
# 1058 1
# 1059 1
# 1060 1
# 1061 1
# 1062 1
# 1063 1
# 1064 1
# 1065 1
# 1066 1
# 1067 1
# 1068 1
# 1069 1
# 1070 1
# 1071 1
# 1072 1
# 1073 1
# 1074 1
# 1075 1
# 1076 1
# 1077 1
# 1078 1
# 1079 1
# 1080 1
# 1081 1
# 1082 1
# 1083 1
# 1084 1
# 1085 1
# 1086 1
# 1087 1
# 1088 1
# 1089 1
# 1090 1
# 1091 1
# 1092 1
# 1093 1
# 1094 1
# 1095 1
# 1096 1
# 1097 1
# 1098 1
# 1099 1
# 1100 1
# 1101 1
# 1102 1
# 1103 1
# 1104 1
# 1105 1
# 1106 1
# 1107 1
# 1108 1
# 1109 1
# 1110 1
# 1111 1
# 1112 1
# 1113 1
# 1114 1
# 1115 1
# 1116 1
# 1117 1
# 1118 1
# 1119 1
# 1120 1
# 1121 1
# 1122 1
# 1123 1
# 1124 1
# 1125 1
# 1126 1
# 1127 1
# 1128 1
# 1129 1
# 1130 1
# 1131 1
# 1132 1
# 1133 1
# 1134 1
# 1135 1
# 1136 1
# 1137 1
# 1138 1
# 1139 1
# 1140 1
# 1141 1
# 1142 1
# 1143 1
# 1144 1
# 1145 1
# 1146 1
# 1147 1
# 1148 1
# 1149 1
# 1150 1
# 1151 1
# 1152 1
# 1153 1
# 1154 1
# 1155 1
# 1156 1
# 1157 1
# 1158 1
# 1159 1
# 1160 1
# 1161 1
# 1162 1
# 1163 1
# 1164 1
# 1165 1
# 1166 1
# 1167 1
# 1168 1
# 1169 1
# 1170 1
# 1171 1
# 1172 1
# 1173 1
# 1174 1
# 1175 1
# 1176 1
# 1177 1
# 1178 1
# 1179 1
# 1180 1
# 1181 1
# 1182 1
# 1183 1
# 1184 1
# 1185 1
# 1186 1
# 1187 1
# 1188 1
# 1189 1
# 1190 1
# 1191 1
# 1192 1
# 1193 1
# 1194 1
# 1195 1
# 1196 1
# 1197 1
# 1198 1
# 1199 1
# 1200 1
# 1201 1
# 1202 1
# 1203 1
# 1204 1
# 1205 1
# 1206 1
# 1207 1
# 1208 1
# 1209 1
# 1210 1
# 1211 1
# 1212 1
# 1213 1
# 1214 1
# 1215 1
# 1216 1
# 1217 1
# 1218 1
# 1219 1
# 1220 1
# 1221 1
# 1222 1
# 1223 1
# 1224 1
# 1225 1
# 1226 1
# 1227 1
# 1228 1
# 1229 1
# 1230 1
# 1231 1
# 1232 1
# 1233 1
# 1234 1
# 1235 1
# 1236 1
# 1237 1
# 1238 1
# 1239 1
# 1240 1
# 1241 1
# 1242 1
# 1243 1
# 1244 1
# 1245 1
# 1246 1
# 1247 1
# 1248 1
# 1249 1
# 1250 1
# 1251 1
# 1252 1
# 1253 1
# 1254 1
# 1255 1
# 1256 1
# 1257 1
# 1258 1
# 1259 1
# 1260 1
# 1261 1
# 1262 1
# 1263 1
# 1264 1
# 1265 1
# 1266 1
# 1267 1
# 1268 1
# 1269 1
# 1270 1
# 1271 1
# 1272 1
# 1273 1
# 1274 1
# 1275 1
# 1276 1
# 1277 1
# 1278 1
# 1279 1
# 1280 1
# 1281 1
# 1282 1
# 1283 1
# 1284 1
# 1285 1
# 1286 1
# 1287 1
# 1288 1
# 1289 1
# 1290 1
# 1291 1
# 1292 1
# 1293 1
# 1294 1
# 1295 1
# 1296 1
# 1297 1
# 1298 1
# 1299 1
# 1300 1
# 1301 1
# 1302 1
# 1303 1
# 1304 1
# 1305 1
# 1306 1
# 1307 1
# 1308 1
# 1309 1
# 1310 1
# 1311 1
# 1312 1
# 1313 1
# 1314 1
# 1315 1
# 1316 1
# 1317 1
# 1318 1
# 1319 1
# 1320 1
# 1321 1
# 1322 1
# 1323 1
# 1324 1
# 1325 1
# 1326 1
# 1327 1
# 1328 1
# 1329 1
# 1330 1
# 1331 1
# 1332 1
# 1333 1
# 1334 1
# 1335 1
# 1336 1
# 1337 1
# 1338 1
# 1339 1
# 1340 1
# 1341 1
# 1342 1
# 1343 1
# 1344 1
# 1345 1
# 1346 1
# 1347 1
# 1348 1
# 1349 1
# 1350 1
# 1351 1
# 1352 1
# 1353 1
# 1354 1
# 1355 1
# 1356 1
# 1357 1
# 1358 1
# 1359 1
# 1360 1
# 1361 1
# 1362 1
# 1363 1
# 1364 1
# 1365 1
# 1366 1
# 1367 1
# 1368 1
# 1369 1
# 1370 1
# 1371 1
# 1372 1
# 1373 1
# 1374 1
# 1375 1
# 1376 1
# 1377 1
# 1378 1
# 1379 1
# 1380 1
# 1381 1
# 1382 1
# 1383 1
# 1384 1
# 1385 1
# 1386 1
# 1387 1
# 1388 1
# 1389 1
# 1390 1
# 1391 1
# 1392 1
# 1393 1
# 1394 1
# 1395 1
# 1396 1
# 1397 1
# 1398 1
# 1399 1
# 1400 1
# 1401 1
# 1402 1
# 1403 1
# 1404 1
# 1405 1
# 1406 1
# 1407 1
# 1408 1
# 1409 1
# 1410 1
# 1411 1
# 1412 1
# 1413 1
# 1414 1
# 1415 1
# 1416 1
# 1417 1
# 1418 1
# 1419 1
# 1420 1
# 1421 1
# 1422 1
# 1423 1
# 1424 1
# 1425 1
# 1426 1
# 1427 1
# 1428 1
# 1429 1
# 1430 1
# 1431 1
# 1432 1
# 1433 1
# 1434 1
# 1435 1
# 1436 1
# 1437 1
# 1438 1
# 1439 1
# 1440 1
# 1441 1
# 1442 1
# 1443 1
# 1444 1
# 1445 1
# 1446 1
# 1447 1
# 1448 1
# 1449 1
# 1450 1
# 1451 1
# 1452 1
# 1453 1
# 1454 1
# 1455 1
# 1456 1
# 1457 1
# 1458 1
# 1459 1
# 1460 1
# 1461 1
# 1462 1
# 1463 1
# 1464 1
# 1465 1
# 1466 1
# 1467 1
# 1468 1
# 1469 1
# 1470 1
# 1471 1
# 1472 1
# 1473 1
# 1474 1
# 1475 1
# 1476 1
# 1477 1
# 1478 1
# 1479 1
# 1480 1
# 1481 1
# 1482 1
# 1483 1
# 1484 1
# 1485 1
# 1486 1
# 1487 1
# 1488 1
# 1489 1
# 1490 1
# 1491 1
# 1492 1
# 1493 1
# 1494 1
# 1495 1
# 1496 1
# 1497 1
# 1498 1
# 1499 1
# 1500 1
# 1501 1
# 1502 1
# 1503 1
# 1504 1
# 1505 1
# 15
```

```
df[column] = LabelEncoder().fit_transform(df[column])
df.head()

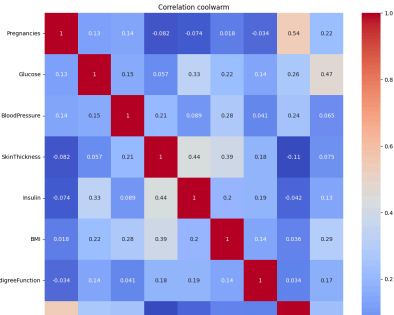
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
0           6       148           72           35           0  33.6           0.627  50           1
1           1       85           66           29           0  26.8           0.351  31           0
2           8       183           66           0           0  23.5           0.672  33           1
3           1       89           66           23  94  28.1           0.167  21           0
4           0       137           40           35  168  43.1           2.288  33           1
```

```
for column in df.select_dtypes(include='float64').columns:
    mean_value = df[column].mean() # Get the mean (most frequent)
    df[column] = df[column].fillna(mean_value)
# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include='object').columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent)
    df[column] = df[column].fillna(mode_value)
df.isnull().sum()
```

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

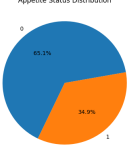
```
sm = heatmap(df_corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Correlation coeefware')
text(0.5, 1.4, 'Correlation coeefware')
```

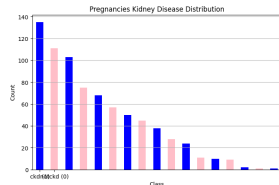


```
appet_counts = df['Outcome'].value_counts()
plt.figure(figsize=(5, 5))
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%',
        startangle=90)
plt.title('Appetite Status Distribution')
plt.show()
```

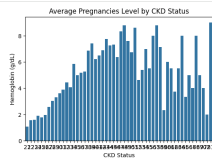
Appetite Status Distribution



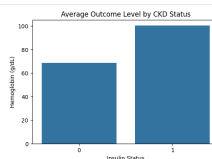
```
plt.figure(figsize=(8, 5))
df['Pregnancies'].value_counts().plot(kind='bar', color='blue',
        label='Pregnancies')
plt.xticks([0, 1], ['0 (1)', '1 (0)'], rotation=45)
plt.title('Pregnancies Kidney Disease Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```



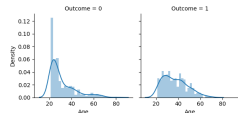
```
import seaborn as sns
plt.figure(figsize=(4, 4))
sns.barplot(x='Age', y='Pregnancies', data=df, estimator='mean',
        ci=None)
plt.title('Average Pregnancies Level by CKD Status')
plt.xlabel('CKD Status')
plt.ylabel('Mean Pregnancies (g/dl)')
plt.show()
```



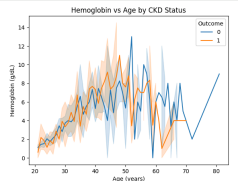
```
import seaborn as sns
plt.figure(figsize=(4, 4))
sns.barplot(x='Outcome', y='Insulin', data=df, estimator='mean',
        ci=None)
plt.title('Average Age Outcome Level by CKD Status')
plt.xlabel('Insulin Status')
plt.ylabel('Mean Hemoglobin (g/dl)')
plt.show()
```



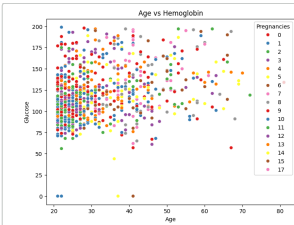
```
g = sns.FacetGrid(df, col='Outcome')
g.map(sns.distplot, 'Age')
plt.show()
```



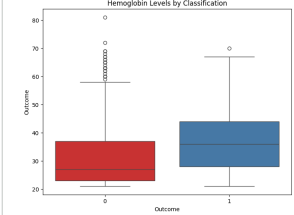
```
sns.lmplot(x='Age', y='Pregnancies', hue='Outcome', data=df)
plt.title('Hemoglobin vs Age by CKD Status')
plt.xlabel('Age (years)')
plt.ylabel('Mean Hemoglobin (g/dl)')
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Age', y='Glucose', data=df, hue='Pregnancies',
        palette='Set1')
plt.title('Age vs Hemoglobin')
plt.xlabel('Age')
plt.ylabel('Glucose')
plt.show()
```

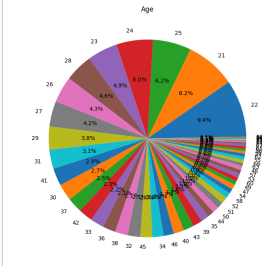


```
plt.figure(figsize=(8, 6))
plt.scatter('Outcome', y='Age', data=df, palette='Set1')
plt.title('Hemoglobin Levels by Classification')
plt.xlabel('Outcome')
plt.ylabel('Outcome')
plt.show()
```



Start coding or [ask a question](#) with AI.

```
grade_counts = df['Age'].value_counts()
plt.figure(figsize=(8, 6))
plt.pie(grade_counts)
plt.title('Age')
plt.show()
```



```
from sklearn.model_selection import GridSearchCV
# Define the Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=0)
# Define the hyperparameters for grid search
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
# Define the evaluation function if it's not already defined
def evaluate_model(model, x_train, y_train, x_test, y_test):
    accuracy = accuracy_score(y_train, model.predict(x_train))
    precision = precision_score(y_train, model.predict(x_train))
    recall = recall_score(y_train, model.predict(x_train))
    f1 = f1_score(y_train, model.predict(x_train))
    report = classification_report(y_train, model.predict(x_train))
    cm = confusion_matrix(y_train, model.predict(x_train))
    return {
        'Model': model.__class__.__name__,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1,
        'Classification Report': report,
        'Confusion Matrix': cm
    }
# Initialize GridSearchCV
grid_search = GridSearchCV(rf_classifier, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
# Fit the grid search to the data
grid_search.fit(x_train, y_train)
# Get the best estimator from grid search
best_rf_classifier = grid_search.best_estimator_
# Make predictions using the best model
y_pred = best_rf_classifier.predict(x_test)
# Evaluate the model
evaluation_results = evaluate_model(best_rf_classifier, x_train, y_train, x_test, y_test)
# Print the evaluation results
for key, value in evaluation_results.items():
    if key in ['Classification Report']:
        print(value)
    elif key in ['Confusion Matrix']:
        print(f'{key}: {value}\n')
    else:
        print(f'{key}: {value}')
# Print the best parameters found by grid search
print(f'Best hyperparameters found by GridSearchCV: {grid_search.best_params_}')
# Print the best parameters found by GridSearchCV
print(f'Best hyperparameters found by GridSearchCV: {grid_search.best_params_}')

Model:
RandomForestClassifier
Accuracy: 0.738
Precision: 0.687
Recall: 0.654
F1 Score: 0.672
precision recall f1-score support
0 0.68 0.78 0.73 55
1 0.62 0.60 0.61 50
accuracy 0.73 0.72 0.73 104
weighted avg 0.73 0.72 0.73 104

Confusion Matrix:
[[27 21]
 [19 31]]

Best hyperparameters found by GridSearchCV:
{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
```

```
from sklearn.feature_selection import SelectFromModel, chi2
# Create selector (choose number of features you want, e.g., k=5)
selector = SelectFromModel(rf_classifier, k=5)
# Fit and transform
selector.fit(x_train, y_train)
print('Selected features:', selector.get_support())
print('Scores:', selector.scores_)
Selected features: [False False True True False False False
False False False False
False True False False False False False False True
True
False False False False False False]
Scores: [1.4688517e-01 1.3807086e-01 2.8130206e-01 1.3993555e-01
1.4688517e-01 1.3807086e-01 1.3712125e-01 1.4688517e-01
1.3712125e-01 1.4688517e-01 1.3712125e-01 1.4688517e-01
1.3712125e-01 1.4688517e-01 1.3712125e-01 1.4688517e-01
1.3712125e-01 1.4688517e-01 1.3712125e-01]
Import pandas as pd
import matplotlib.pyplot as plt
# Create DataFrame of features and chi-square scores
data = pd.DataFrame({'feature': x.columns, 'score': selector.scores_})
# Sort by score (ascending), but make plot (descending)
data = data.sort_values('score', ascending=False)
# Plot bar chart
plt.figure(figsize=(10, 6))
plt.bar(data['feature'], data['score'])
plt.xlabel('Feature')
plt.ylabel('Chi-Square Score')
plt.title('Chi-Square Feature Selection Scores')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

File: /tmp/option-input-VN6A8L73.py, line 6
Selected features: [False False True True False False False
SyntaxError: invalid syntax
```

Next steps: [Continue](#)

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Get feature importance from the best Random Forest classifier
feature_importance = best_rf_classifier.feature_importances_

# Get feature names from the training data (X)
feature_names = x.columns

# Create a DataFrame for better visualization
importance_df = pd.DataFrame({
    'feature': feature_names,
    'importance': feature_importance
})

# Sort the DataFrame by importance in descending order
importance_df = importance_df.sort_values('importance', ascending=False)

# Plot the feature importance
plt.figure(figsize=(10, 6))
plt.bar(importance_df['feature'], importance_df['importance'])
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

