{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"id": "ca9391d7-b458-463d-b9ca-b4a182ccfac6",

"metadata": {},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"\n",

"KeyboardInterrupt\n",

"\n"

]

}

],

"source": [

"# The receiver-operating characteristic curves for ten ML models without undergoing SMOTE oversampling processing\n",

"import pandas as pd\n",

"from sklearn.preprocessing import OneHotEncoder, StandardScaler\n",

"from sklearn.compose import ColumnTransformer\n",

"from imblearn.over\_sampling import SMOTE\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier\n",

"from sklearn.tree import DecisionTreeClassifier\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.naive\_bayes import GaussianNB\n",

"from sklearn.neural\_network import MLPClassifier\n",

"from sklearn.metrics import roc\_curve, roc\_auc\_score\n",

"import matplotlib.pyplot as plt\n",

"import lightgbm as lgb\n",

"import xgboost as xgb\n",

"from catboost import CatBoostClassifier\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"from sklearn.pipeline import Pipeline\n",

"from sklearn.model\_selection import cross\_val\_score\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"from sklearn.pipeline import Pipeline\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"from sklearn.pipeline import Pipeline\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"from sklearn.pipeline import Pipeline\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"from sklearn.pipeline import Pipeline\n",

"\n",

"\n",

"file\_path = r'D:\\2024\_4\_29.xlsx'\n",

"data = pd.read\_excel(file\_path)\n",

"\n",

"\n",

"features = ['性别', '糖尿病', '年龄', '白细胞计数结果标志', '单核细胞计数结果标志', '红细胞结果标志', \n",

" '淋巴细胞计数结果标志', '血红蛋白结果标志', '血小板计数结果标志', '葡萄糖结果标志', \n",

" '白蛋白结果标志', '总胆红素结果标志', \"C反应蛋白结果标志\", \"RBC体积分布宽度结果标志\", \n",

" \"红细胞压积结果标志\", \"平均RBC血红蛋白含量结果标志\", \"平均RBC血红蛋白浓度结果标志\", \n",

" \"平均红细胞体积结果标志\", 'Braden评分（压疮评分）', 'NRS2002评分（营养风险筛查）', \n",

" 'GlasGow评分（昏迷评分）', 'CPOT评分（疼痛评分表）', '误吸/窒息评分', '肠内营养耐受性', \n",

" 'VTE评分（静脉血栓栓塞症风险）', '非计划拔管评估']\n",

"target = '院感'\n",

"\n",

"X = data[features]\n",

"y = data[target]\n",

"\n",

"\n",

"categorical\_features = ['白细胞计数结果标志', '单核细胞计数结果标志', '红细胞结果标志', \n",

" '淋巴细胞计数结果标志', '血红蛋白结果标志', '血小板计数结果标志', '葡萄糖结果标志', \n",

" '白蛋白结果标志', '总胆红素结果标志', \"C反应蛋白结果标志\", \"RBC体积分布宽度结果标志\", \n",

" \"红细胞压积结果标志\", \"平均RBC血红蛋白含量结果标志\", \"平均RBC血红蛋白浓度结果标志\", \n",

" \"平均红细胞体积结果标志\"]\n",

"categorical\_transformer = OneHotEncoder()\n",

"\n",

"\n",

"numeric\_features = ['年龄', 'Braden评分（压疮评分）', 'NRS2002评分（营养风险筛查）', 'GlasGow评分（昏迷评分）',\n",

" 'CPOT评分（疼痛评分表）', '误吸/窒息评分', '肠内营养耐受性', 'VTE评分（静脉血栓栓塞症风险）', '非计划拔管评估']\n",

"numeric\_transformer = StandardScaler()\n",

"\n",

"preprocessor = ColumnTransformer(\n",

" transformers=[\n",

" ('num', numeric\_transformer, numeric\_features),\n",

" ('cat', categorical\_transformer, categorical\_features)\n",

" ])\n",

"\n",

"\n",

"pipeline = Pipeline(steps=[('preprocessor', preprocessor),\n",

" ('feature\_selection', SelectFromModel(LogisticRegression(penalty=\"l1\", solver='liblinear')))])\n",

"\n",

"\n",

"X\_selected = pipeline.fit\_transform(X, y)\n",

"\n",

"\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)\n",

"\n",

"\n",

"classifiers = {\n",

" \"Random Forest\": RandomForestClassifier(),\n",

" \"XGBoost\": xgb.XGBClassifier(),\n",

" \"MLP\": MLPClassifier(),\n",

" \"LightGBM\": lgb.LGBMClassifier(),\n",

" \"Logistic Regression\": LogisticRegression(),\n",

" \"Decision tree\": DecisionTreeClassifier(),\n",

" \"Naive bayes\": GaussianNB(),\n",

" \"AdaBoost\": AdaBoostClassifier(),\n",

" \"CatBoost\": CatBoostClassifier(verbose=False),\n",

" \"Neural Networks\": MLPClassifier()\n",

"}\n",

"\n",

"\n",

"results = {}\n",

"for name, clf in classifiers.items():\n",

" clf.fit(X\_train, y\_train)\n",

" y\_pred = clf.predict\_proba(X\_test)[:, 1]\n",

" fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)\n",

" auc = roc\_auc\_score(y\_test, y\_pred)\n",

" results[name] = {\"fpr\": fpr, \"tpr\": tpr, \"auc\": auc}\n",

"\n",

"\n",

"plt.figure(figsize=(10, 8))\n",

"\n",

"sorted\_results = sorted(results.items(), key=lambda x: x[1][\"auc\"], reverse=True)\n",

"for name, result in sorted\_results:\n",

" plt.plot(result[\"fpr\"], result[\"tpr\"], label=f'{name} (AUC = {result[\"auc\"]:.3f})')\n",

"plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guessing')\n",

"plt.xlabel('False Positive Rate')\n",

"plt.ylabel('True Positive Rate')\n",

"plt.title('Receiver Operating Characteristic (ROC) Curve')\n",

"plt.legend(loc='lower right')\n",

"plt.grid(True)\n",

"plt.show()\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "2dc82626-dc96-4916-9187-6ce9f0caa38b",

"metadata": {},

"outputs": [],

"source": [

"# The ten-fold cross-validation results of the CatBoost model without SMOTE oversampling.\n",

"from sklearn.model\_selection import cross\_val\_predict\n",

"from sklearn.model\_selection import StratifiedKFold\n",

"from sklearn import metrics\n",

"import numpy as np\n",

"\n",

"\n",

"clf = CatBoostClassifier(verbose=False)\n",

"\n",

"\n",

"cv = StratifiedKFold(n\_splits=10, shuffle=True, random\_state=42)\n",

"\n",

"\n",

"fold\_results = []\n",

"\n",

"\n",

"for i, (train\_index, test\_index) in enumerate(cv.split(X\_selected, y)):\n",

" X\_train, X\_test = X\_selected[train\_index], X\_selected[test\_index]\n",

" y\_train, y\_test = y[train\_index], y[test\_index]\n",

" \n",

" clf.fit(X\_train, y\_train)\n",

" y\_prob = clf.predict\_proba(X\_test)[:, 1]\n",

" fpr, tpr, \_ = metrics.roc\_curve(y\_test, y\_prob)\n",

" roc\_auc = metrics.auc(fpr, tpr)\n",

" fold\_results.append({\"fpr\": fpr, \"tpr\": tpr, \"auc\": roc\_auc, \"fold\": i+1})\n",

"\n",

"tprs = []\n",

"aucs = []\n",

"mean\_fpr = np.linspace(0, 1, 100)\n",

"plt.figure(figsize=(10, 8))\n",

"\n",

"for result in fold\_results:\n",

" plt.plot(result[\"fpr\"], result[\"tpr\"], alpha=0.3, label=f'Fold {result[\"fold\"]} (AUC = {result[\"auc\"]:.3f})')\n",

" interp\_tpr = np.interp(mean\_fpr, result[\"fpr\"], result[\"tpr\"])\n",

" interp\_tpr[0] = 0.0\n",

" tprs.append(interp\_tpr)\n",

" aucs.append(result[\"auc\"])\n",

"\n",

"mean\_tpr = np.mean(tprs, axis=0)\n",

"mean\_tpr[-1] = 1.0\n",

"mean\_auc = metrics.auc(mean\_fpr, mean\_tpr)\n",

"\n",

"std\_auc = np.std(aucs)\n",

"\n",

"tpr\_upper = np.minimum(mean\_tpr + 1.96 \* np.std(tprs, axis=0), 1)\n",

"tpr\_lower = np.maximum(mean\_tpr - 1.96 \* np.std(tprs, axis=0), 0)\n",

"\n",

"plt.plot(mean\_fpr, mean\_tpr, color='b', label=f'Mean ROC (AUC = {mean\_auc:.3f} $\\pm$ {std\_auc:.3f})')\n",

"plt.fill\_between(mean\_fpr, tpr\_lower, tpr\_upper, color='grey', alpha=0.3)\n",

"\n",

"plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guessing')\n",

"plt.xlabel('False Positive Rate')\n",

"plt.ylabel('True Positive Rate')\n",

"plt.title('Receiver Operating Characteristic (ROC) Curve')\n",

"plt.legend(loc='lower right')\n",

"plt.grid(True)\n",

"\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "311066ad-8866-46b4-bab9-df0b2a1922d7",

"metadata": {},

"outputs": [],

"source": [

"# The receiver-operating characteristic curves for ten ML models with undergoing SMOTE oversampling processing\n",

"import pandas as pd\n",

"from sklearn.preprocessing import OneHotEncoder, StandardScaler\n",

"from sklearn.compose import ColumnTransformer\n",

"from imblearn.over\_sampling import SMOTE\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier\n",

"from sklearn.tree import DecisionTreeClassifier\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.naive\_bayes import GaussianNB\n",

"from sklearn.neural\_network import MLPClassifier\n",

"from sklearn.metrics import roc\_curve, roc\_auc\_score\n",

"import matplotlib.pyplot as plt\n",

"import lightgbm as lgb\n",

"import xgboost as xgb\n",

"from catboost import CatBoostClassifier\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"from sklearn.pipeline import Pipeline\n",

"\n",

"\n",

"file\_path = r'D:\\2024\_4\_29.xlsx'\n",

"data = pd.read\_excel(file\_path)\n",

"\n",

"\n",

"features = ['性别', '糖尿病', '年龄', '白细胞计数结果标志', '单核细胞计数结果标志', '红细胞结果标志', \n",

" '淋巴细胞计数结果标志', '血红蛋白结果标志', '血小板计数结果标志', '葡萄糖结果标志', \n",

" '白蛋白结果标志', '总胆红素结果标志', \"C反应蛋白结果标志\", \"RBC体积分布宽度结果标志\", \n",

" \"红细胞压积结果标志\", \"平均RBC血红蛋白含量结果标志\", \"平均RBC血红蛋白浓度结果标志\", \n",

" \"平均红细胞体积结果标志\", 'Braden评分（压疮评分）', 'NRS2002评分（营养风险筛查）', \n",

" 'GlasGow评分（昏迷评分）', 'CPOT评分（疼痛评分表）', '误吸/窒息评分', '肠内营养耐受性', \n",

" 'VTE评分（静脉血栓栓塞症风险）', '非计划拔管评估']\n",

"target = '院感'\n",

"\n",

"X = data[features]\n",

"y = data[target]\n",

"\n",

"\n",

"categorical\_features = ['白细胞计数结果标志', '单核细胞计数结果标志', '红细胞结果标志', \n",

" '淋巴细胞计数结果标志', '血红蛋白结果标志', '血小板计数结果标志', '葡萄糖结果标志', \n",

" '白蛋白结果标志', '总胆红素结果标志', \"C反应蛋白结果标志\", \"RBC体积分布宽度结果标志\", \n",

" \"红细胞压积结果标志\", \"平均RBC血红蛋白含量结果标志\", \"平均RBC血红蛋白浓度结果标志\", \n",

" \"平均红细胞体积结果标志\"]\n",

"categorical\_transformer = OneHotEncoder()\n",

"\n",

"\n",

"numeric\_features = ['年龄', 'Braden评分（压疮评分）', 'NRS2002评分（营养风险筛查）', 'GlasGow评分（昏迷评分）',\n",

" 'CPOT评分（疼痛评分表）', '误吸/窒息评分', '肠内营养耐受性', 'VTE评分（静脉血栓栓塞症风险）', '非计划拔管评估']\n",

"\n",

"numeric\_transformer = StandardScaler()\n",

"\n",

"\n",

"preprocessor = ColumnTransformer(\n",

" transformers=[\n",

" ('num', numeric\_transformer, numeric\_features),\n",

" ('cat', categorical\_transformer, categorical\_features)\n",

" ])\n",

"\n",

"\n",

"X\_preprocessed = preprocessor.fit\_transform(X)\n",

"\n",

"\n",

"smote = SMOTE(random\_state=42)\n",

"X\_resampled, y\_resampled = smote.fit\_resample(X\_preprocessed, y)\n",

"\n",

"\n",

"selector = SelectFromModel(LogisticRegression(penalty=\"l1\", solver='liblinear'))\n",

"X\_selected = selector.fit\_transform(X\_resampled, y\_resampled)\n",

"\n",

"\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y\_resampled, test\_size=0.2, random\_state=42)\n",

"\n",

"\n",

"classifiers = {\n",

" \"Random Forest\": RandomForestClassifier(),\n",

" \"XGBoost\": xgb.XGBClassifier(),\n",

" \"MLP\": MLPClassifier(),\n",

" \"LightGBM\": lgb.LGBMClassifier(),\n",

" \"Logistic Regression\": LogisticRegression(),\n",

" \"Decision tree\": DecisionTreeClassifier(),\n",

" \"Naive bayes\": GaussianNB(),\n",

" \"AdaBoost\": AdaBoostClassifier(),\n",

" \"CatBoost\": CatBoostClassifier(verbose=False),\n",

" \"Neural Networks\": MLPClassifier()\n",

"}\n",

"\n",

"\n",

"results = {}\n",

"for name, clf in classifiers.items():\n",

" clf.fit(X\_train, y\_train)\n",

" y\_pred = clf.predict\_proba(X\_test)[:, 1]\n",

" fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)\n",

" auc = roc\_auc\_score(y\_test, y\_pred)\n",

" results[name] = {\"fpr\": fpr, \"tpr\": tpr, \"auc\": auc}\n",

"\n",

"\n",

"sorted\_results = sorted(results.items(), key=lambda x: x[1][\"auc\"], reverse=True)\n",

"\n",

"\n",

"plt.figure(figsize=(10, 8))\n",

"for name, result in sorted\_results:\n",

" plt.plot(result[\"fpr\"], result[\"tpr\"], label=f'{name} (AUC = {result[\"auc\"]:.3f})')\n",

"plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guessing')\n",

"plt.xlabel('False Positive Rate')\n",

"plt.ylabel('True Positive Rate')\n",

"plt.title('Receiver Operating Characteristic (ROC) Curve')\n",

"plt.legend(loc='lower right')\n",

"plt.grid(True)\n",

"plt.show()\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "e1679ce0-2224-47fe-b52f-ae55b13f1976",

"metadata": {},

"outputs": [],

"source": [

"# The ten-fold cross-validation results of the CatBoost model with SMOTE oversampling\n",

"import pandas as pd\n",

"from sklearn.preprocessing import OneHotEncoder, StandardScaler\n",

"from sklearn.compose import ColumnTransformer\n",

"from imblearn.over\_sampling import SMOTE\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier\n",

"from sklearn.tree import DecisionTreeClassifier\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.naive\_bayes import GaussianNB\n",

"from sklearn.neural\_network import MLPClassifier\n",

"from sklearn.metrics import roc\_curve, roc\_auc\_score\n",

"import matplotlib.pyplot as plt\n",

"import lightgbm as lgb\n",

"import xgboost as xgb\n",

"from catboost import CatBoostClassifier\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"from sklearn.pipeline import Pipeline\n",

"\n",

"\n",

"file\_path = r'D:\\2024\_4\_29.xlsx'\n",

"data = pd.read\_excel(file\_path)\n",

"\n",

"features = ['性别', '糖尿病', '年龄', '白细胞计数结果标志', '单核细胞计数结果标志', '红细胞结果标志', \n",

" '淋巴细胞计数结果标志', '血红蛋白结果标志', '血小板计数结果标志', '葡萄糖结果标志', \n",

" '白蛋白结果标志', '总胆红素结果标志', \"C反应蛋白结果标志\", \"RBC体积分布宽度结果标志\", \n",

" \"红细胞压积结果标志\", \"平均RBC血红蛋白含量结果标志\", \"平均RBC血红蛋白浓度结果标志\", \n",

" \"平均红细胞体积结果标志\", 'Braden评分（压疮评分）', 'NRS2002评分（营养风险筛查）', \n",

" 'GlasGow评分（昏迷评分）', 'CPOT评分（疼痛评分表）', '误吸/窒息评分', '肠内营养耐受性', \n",

" 'VTE评分（静脉血栓栓塞症风险）', '非计划拔管评估',\"嗜碱性粒细胞计数结果标志\",\n",

" \"嗜酸性粒细胞计数结果标志\",\"血小板分布宽度结果标志\",\"血小板压积结果标志\",\"中性粒细胞计数结果标志\"]\n",

"target = '院感'\n",

"\n",

"X = data[features]\n",

"y = data[target]\n",

"\n",

"\n",

"categorical\_features = ['白细胞计数结果标志', '单核细胞计数结果标志', '红细胞结果标志', \n",

" '淋巴细胞计数结果标志', '血红蛋白结果标志', '血小板计数结果标志', '葡萄糖结果标志', \n",

" '白蛋白结果标志', '总胆红素结果标志', \"C反应蛋白结果标志\", \"RBC体积分布宽度结果标志\", \n",

" \"红细胞压积结果标志\", \"平均RBC血红蛋白含量结果标志\", \"平均RBC血红蛋白浓度结果标志\", \n",

" \"平均红细胞体积结果标志\",\"嗜碱性粒细胞计数结果标志\",\"嗜酸性粒细胞计数结果标志\",\n",

" \"血小板分布宽度结果标志\",\"血小板压积结果标志\",\"中性粒细胞计数结果标志\"]\n",

"categorical\_transformer = OneHotEncoder()\n",

"\n",

"\n",

"numeric\_features = ['年龄', 'Braden评分（压疮评分）', 'NRS2002评分（营养风险筛查）', 'GlasGow评分（昏迷评分）',\n",

" 'CPOT评分（疼痛评分表）', '误吸/窒息评分', '肠内营养耐受性', 'VTE评分（静脉血栓栓塞症风险）', '非计划拔管评估']\n",

"numeric\_transformer = StandardScaler()\n",

"\n",

"\n",

"preprocessor = ColumnTransformer(\n",

" transformers=[\n",

" ('num', numeric\_transformer, numeric\_features),\n",

" ('cat', categorical\_transformer, categorical\_features)\n",

" ])\n",

"\n",

"\n",

"X\_preprocessed = preprocessor.fit\_transform(X)\n",

"\n",

"\n",

"smote = SMOTE(random\_state=42)\n",

"X\_resampled, y\_resampled = smote.fit\_resample(X\_preprocessed, y)\n",

"\n",

"selector = SelectFromModel(LogisticRegression(penalty=\"l1\", solver='liblinear'))\n",

"X\_selected = selector.fit\_transform(X\_resampled, y\_resampled)\n",

"\n",

"\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y\_resampled, test\_size=0.2, random\_state=42)\n",

"\n",

"\n",

"classifiers = {\n",

" \"Random Forest\": RandomForestClassifier(),\n",

" \"XGBoost\": xgb.XGBClassifier(),\n",

" \"MLP\": MLPClassifier(),\n",

" \"LightGBM\": lgb.LGBMClassifier(),\n",

" \"Logistic Regression\": LogisticRegression(),\n",

" \"Decision tree\": DecisionTreeClassifier(),\n",

" \"Naive bayes\": GaussianNB(),\n",

" \"AdaBoost\": AdaBoostClassifier(),\n",

" \"CatBoost\": CatBoostClassifier(verbose=False),\n",

" \"Neural Networks\": MLPClassifier()\n",

"}\n",

"\n",

"\n",

"results = {}\n",

"for name, clf in classifiers.items():\n",

" clf.fit(X\_train, y\_train)\n",

" y\_pred = clf.predict\_proba(X\_test)[:, 1]\n",

" fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)\n",

" auc = roc\_auc\_score(y\_test, y\_pred)\n",

" results[name] = {\"fpr\": fpr, \"tpr\": tpr, \"auc\": auc}\n",

"\n",

"\n",

"sorted\_results = sorted(results.items(), key=lambda x: x[1][\"auc\"], reverse=True)\n",

"\n",

"\n",

"plt.figure(figsize=(10, 8))\n",

"catboost\_result = results[\"CatBoost\"]\n",

"plt.plot(catboost\_result[\"fpr\"], catboost\_result[\"tpr\"], label=f'CatBoost (AUC = {catboost\_result[\"auc\"]:.3f})')\n",

"plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guessing')\n",

"plt.xlabel('False Positive Rate')\n",

"plt.ylabel('True Positive Rate')\n",

"plt.title('Receiver Operating Characteristic (ROC) Curve for CatBoost')\n",

"plt.legend(loc='lower right')\n",

"plt.grid(True)\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "6161c23c-1c36-4d2b-820a-6f7929e6f428",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "752e30be-d5a0-4a0e-903a-83f68944c870",

"metadata": {},

"outputs": [],

"source": [

"# SHapley Additive exPlanations\n",

"import pandas as pd\n",

"from imblearn.over\_sampling import SMOTE\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier\n",

"from sklearn.tree import DecisionTreeClassifier\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.naive\_bayes import GaussianNB\n",

"from sklearn.neural\_network import MLPClassifier\n",

"from sklearn.metrics import roc\_curve, roc\_auc\_score\n",

"import matplotlib.pyplot as plt\n",

"import lightgbm as lgb\n",

"import xgboost as xgb\n",

"from catboost import CatBoostClassifier\n",

"from sklearn.feature\_selection import SelectFromModel\n",

"\n",

"\n",

"file\_path = r'D:\\2024\_5\_23.xlsx'\n",

"data = pd.read\_excel(file\_path)\n",

"\n",

"\n",

"features = ['Age','Braden score','NRS 2002 score','GCS score','PAS','Enteral feeding tolerance',\n",

"'CRS','Female','Male','High WBC count','Low WBC count','Medium WBC count',\n",

"'High RBC','Low RBC','Medium RBC','High lymphocyte count','Low lymphocyte count','Medium lymphocyte count',\n",

"'High hemoglobin level','Low hemoglobin level','Medium hemoglobin level','High total bilirubin level','Low total bilirubin level',\n",

"'Medium total bilirubin level','High RBC distribution width','Low RBC distribution width','Medium RBC distribution width','High hematocrit',\n",

"'Low hematocrit','Medium hematocrit','High mean corpuscular hemoglobin concentration','Low mean corpuscular hemoglobin concentration',\n",

"'Medium mean corpuscular hemoglobin concentration','High mean corpuscular volume','Low mean corpuscular volume','Medium mean corpuscular volume',\n",

"'High eosinophil count','low eosinophil count','Medium eosinophil count']\n",

"\n",

"target = 'HAI'\n",

"\n",

"X = data[features]\n",

"y = data[target]\n",

"\n",

"\n",

"smote = SMOTE(random\_state=42)\n",

"X\_resampled, y\_resampled = smote.fit\_resample(X, y)\n",

"\n",

"\n",

"selector = SelectFromModel(LogisticRegression(penalty=\"l1\", solver='liblinear'))\n",

"X\_selected = selector.fit\_transform(X\_resampled, y\_resampled)\n",

"\n",

"\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y\_resampled, test\_size=0.2, random\_state=42)\n",

"\n",

"\n",

"classifiers = {\n",

" \"Random Forest\": RandomForestClassifier(),\n",

" \"XGBoost\": xgb.XGBClassifier(),\n",

" \"MLP\": MLPClassifier(),\n",

" \"LightGBM\": lgb.LGBMClassifier(),\n",

" \"Logistic Regression\": LogisticRegression(),\n",

" \"Decision tree\": DecisionTreeClassifier(),\n",

" \"Naive bayes\": GaussianNB(),\n",

" \"AdaBoost\": AdaBoostClassifier(),\n",

" \"CatBoost\": CatBoostClassifier(verbose=False),\n",

" \"Neural Networks\": MLPClassifier()\n",

"}\n",

"\n",

"\n",

"results = {}\n",

"for name, clf in classifiers.items():\n",

" clf.fit(X\_train, y\_train)\n",

" y\_pred = clf.predict\_proba(X\_test)[:, 1]\n",

" fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)\n",

" auc = roc\_auc\_score(y\_test, y\_pred)\n",

" results[name] = {\"fpr\": fpr, \"tpr\": tpr, \"auc\": auc}\n",

"\n",

"\n",

"sorted\_results = sorted(results.items(), key=lambda x: x[1][\"auc\"], reverse=True)\n",

"\n",

"\n",

"plt.figure(figsize=(10, 8))\n",

"for name, result in sorted\_results:\n",

" plt.plot(result[\"fpr\"], result[\"tpr\"], label=f'{name} (AUC = {result[\"auc\"]:.3f})')\n",

"plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guessing')\n",

"plt.xlabel('False Positive Rate')\n",

"plt.ylabel('True Positive Rate')\n",

"plt.title('Receiver Operating Characteristic (ROC) Curve')\n",

"plt.legend(loc='lower right')\n",

"plt.grid(True)\n",

"plt.show()\n",

"\n",

"import shap\n",

"import matplotlib.pyplot as plt\n",

"\n",

"\n",

"feature\_names = features\n",

"\n",

"\n",

"explainer = shap.TreeExplainer(classifiers['CatBoost']) \n",

"shap\_values = explainer.shap\_values(X\_test)\n",

"\n",

"\n",

"shap.summary\_plot(shap\_values, X\_test, feature\_names=feature\_names, plot\_type='violin', show=False)\n",

"plt.gcf().set\_size\_inches(20, 8) \n",

"\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "09e10c54-e8c0-426e-9031-9f63831cb5dc",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.8.10"

}

},

"nbformat": 4,

"nbformat\_minor": 5

"cells": [

{

"cell\_type": "code",

"execution\_count": null,

"id": "b3ba9dab-c635-4378-9f3f-b67850c1b733",

"metadata": {},

"outputs": [],

"source": [

"from sklearn.metrics import roc\_curve, roc\_auc\_score, confusion\_matrix\n",

"import numpy as np\n",

"\n",

"\n",

"catboost\_clf = CatBoostClassifier(verbose=False)\n",

"catboost\_clf.fit(X\_train, y\_train)\n",

"\n",

"\n",

"y\_pred\_proba = catboost\_clf.predict\_proba(X\_test)[:, 1]\n",

"\n",

"\n",

"fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_proba)\n",

"\n",

"\n",

"youden\_j = tpr - fpr\n",

"optimal\_idx = np.argmax(youden\_j)\n",

"optimal\_threshold = thresholds[optimal\_idx]\n",

"\n",

"\n",

"print(f\"Optimal threshold for CatBoost model: {optimal\_threshold}\")\n",

"\n",

"\n",

"y\_pred\_optimal = (y\_pred\_proba >= optimal\_threshold).astype(int)\n",

"cm = confusion\_matrix(y\_test, y\_pred\_optimal)\n",

"print(\"Confusion Matrix at optimal threshold:\")\n",

"print(cm)"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.8.10"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}