```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "id": "UPxkew1vJT0K"
      },
      "outputs": [],
      "source": [
        "import pandas as pd\n",
        "data = pd.read_csv(\"/content/Creditcard_data.csv\" )"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "data.head()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 236
        },
        "id": "XKeMP156JpjQ",
        "outputId": "8a7282b7-2337-4dbb-ffdc-07ee1a22b627"
      },
      "execution_count": 2,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "   Time        V1        V2        V3        V4        V5
V6        V7  \\\n",
              "0     0 -1.359807 -0.072781  2.536347  1.378155 -0.338321
0.462388  0.239599   \n",
              "1     0  1.191857  0.266151  0.166480  0.448154  0.060018
-0.082361 -0.078803   \n",
              "2     1 -1.358354 -1.340163  1.773209  0.379780 -0.503198
1.800499  0.791461   \n",
              "3     1 -0.966272 -0.185226  1.792993 -0.863291 -0.010309
1.247203  0.237609   \n",
              "4     2 -1.158233  0.877737  1.548718  0.403034 -0.407193
0.095921  0.592941   \n",
              "\n",
              "         V8        V9  ...       V21       V22       V23
V24       V25  \\\n",
```

```
            "0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474
0.066928  0.128539   \n",
            "1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288
-0.339846  0.167170   \n",
            "2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412
-0.689281 -0.327642   \n",
            "3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321
-1.175575  0.647376   \n",
            "4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458
0.141267 -0.206010   \n",
            "\n",
            "        V26       V27       V28  Amount  Class  \n",
            "0 -0.189115  0.133558 -0.021053  149.62      0  \n",
            "1  0.125895 -0.008983  0.014724    2.69      1  \n",
            "2 -0.139097 -0.055353 -0.059752  378.66      0  \n",
            "3 -0.221929  0.062723  0.061458  123.50      0  \n",
            "4  0.502292  0.219422  0.215153   69.99      0  \n",
            "\n",
            "[5 rows x 31 columns]"
          ],
          "text/html": [
            "\n",
            "  <div id=\"df-d458d352-7bbe-4a6f-85d7-673ac43ce016\"
class=\"colab-df-container\">\n",
            "    <div>\n",
            "<style scoped>\n",
            "    .dataframe tbody tr th:only-of-type {\n",
            "        vertical-align: middle;\n",
            "    }\n",
            "\n",
            "    .dataframe tbody tr th {\n",
            "        vertical-align: top;\n",
            "    }\n",
            "\n",
            "    .dataframe thead th {\n",
            "        text-align: right;\n",
            "    }\n",
            "</style>\n",
            "<table border=\"1\" class=\"dataframe\">\n",
            "  <thead>\n",
            "    <tr style=\"text-align: right;\">\n",
            "      <th></th>\n",
            "      <th>Time</th>\n",
            "      <th>V1</th>\n",
            "      <th>V2</th>\n",
            "      <th>V3</th>\n",
            "      <th>V4</th>\n",
            "      <th>V5</th>\n",
            "      <th>V6</th>\n",
            "      <th>V7</th>\n",
            "      <th>V8</th>\n",
            "      <th>V9</th>\n",
            "      <th>...</th>\n",
            "      <th>V21</th>\n",
            "      <th>V22</th>\n",
            "      <th>V23</th>\n",
            "      <th>V24</th>\n",
            "      <th>V25</th>\n",
            "      <th>V26</th>\n",
            "      <th>V27</th>\n",
            "      <th>V28</th>\n",
            "      <th>Amount</th>\n",
```

```
"      <th>Class</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>0</td>\n",
"      <td>-1.359807</td>\n",
"      <td>-0.072781</td>\n",
"      <td>2.536347</td>\n",
"      <td>1.378155</td>\n",
"      <td>-0.338321</td>\n",
"      <td>0.462388</td>\n",
"      <td>0.239599</td>\n",
"      <td>0.098698</td>\n",
"      <td>0.363787</td>\n",
"      <td>...</td>\n",
"      <td>-0.018307</td>\n",
"      <td>0.277838</td>\n",
"      <td>-0.110474</td>\n",
"      <td>0.066928</td>\n",
"      <td>0.128539</td>\n",
"      <td>-0.189115</td>\n",
"      <td>0.133558</td>\n",
"      <td>-0.021053</td>\n",
"      <td>149.62</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>0</td>\n",
"      <td>1.191857</td>\n",
"      <td>0.266151</td>\n",
"      <td>0.166480</td>\n",
"      <td>0.448154</td>\n",
"      <td>0.060018</td>\n",
"      <td>-0.082361</td>\n",
"      <td>-0.078803</td>\n",
"      <td>0.085102</td>\n",
"      <td>-0.255425</td>\n",
"      <td>...</td>\n",
"      <td>-0.225775</td>\n",
"      <td>-0.638672</td>\n",
"      <td>0.101288</td>\n",
"      <td>-0.339846</td>\n",
"      <td>0.167170</td>\n",
"      <td>0.125895</td>\n",
"      <td>-0.008983</td>\n",
"      <td>0.014724</td>\n",
"      <td>2.69</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>1</td>\n",
"      <td>-1.358354</td>\n",
"      <td>-1.340163</td>\n",
"      <td>1.773209</td>\n",
"      <td>0.379780</td>\n",
"      <td>-0.503198</td>\n",
"      <td>1.800499</td>\n",
"      <td>0.791461</td>\n",
```

```
"      <td>0.247676</td>\n",
"      <td>-1.514654</td>\n",
"      <td>...</td>\n",
"      <td>0.247998</td>\n",
"      <td>0.771679</td>\n",
"      <td>0.909412</td>\n",
"      <td>-0.689281</td>\n",
"      <td>-0.327642</td>\n",
"      <td>-0.139097</td>\n",
"      <td>-0.055353</td>\n",
"      <td>-0.059752</td>\n",
"      <td>378.66</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>1</td>\n",
"      <td>-0.966272</td>\n",
"      <td>-0.185226</td>\n",
"      <td>1.792993</td>\n",
"      <td>-0.863291</td>\n",
"      <td>-0.010309</td>\n",
"      <td>1.247203</td>\n",
"      <td>0.237609</td>\n",
"      <td>0.377436</td>\n",
"      <td>-1.387024</td>\n",
"      <td>...</td>\n",
"      <td>-0.108300</td>\n",
"      <td>0.005274</td>\n",
"      <td>-0.190321</td>\n",
"      <td>-1.175575</td>\n",
"      <td>0.647376</td>\n",
"      <td>-0.221929</td>\n",
"      <td>0.062723</td>\n",
"      <td>0.061458</td>\n",
"      <td>123.50</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>2</td>\n",
"      <td>-1.158233</td>\n",
"      <td>0.877737</td>\n",
"      <td>1.548718</td>\n",
"      <td>0.403034</td>\n",
"      <td>-0.407193</td>\n",
"      <td>0.095921</td>\n",
"      <td>0.592941</td>\n",
"      <td>-0.270533</td>\n",
"      <td>0.817739</td>\n",
"      <td>...</td>\n",
"      <td>-0.009431</td>\n",
"      <td>0.798278</td>\n",
"      <td>-0.137458</td>\n",
"      <td>0.141267</td>\n",
"      <td>-0.206010</td>\n",
"      <td>0.502292</td>\n",
"      <td>0.219422</td>\n",
"      <td>0.215153</td>\n",
"      <td>69.99</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
```

```
            "  </tbody>\n",
            "</table>\n",
            "<p>5 rows × 31 columns</p>\n",
            "</div>\n",
            "    <div class=\"colab-df-buttons\">\n",
            "\n",
            "  <div class=\"colab-df-container\">\n",
            "    <button class=\"colab-df-convert\"
onclick=\"convertToInteractive('df-d458d352-7bbe-4a6f-85d7-673ac43ce016')\"\n",
            "            title=\"Convert this dataframe to an interactive
table.\"\n",
            "            style=\"display:none;\">\n",
            "\n",
            "  <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"
viewBox=\"0 -960 960 960\">\n",
            "    <path d=\"M120-120v-720h720v720H120Zm60-500h600v-
160H180v160Zm220 220h160v-160H400v160Zm0 220h160v-160H400v160ZM180-400h160v-
160H180v160Zm440 0h160v-160H620v160ZM180-180h160v-160H180v160Zm440 0h160v-
160H620v160Z\"/>\n",
            "  </svg>\n",
            "    </button>\n",
            "\n",
            "  <style>\n",
            "    .colab-df-container {\n",
            "      display:flex;\n",
            "      gap: 12px;\n",
            "    }\n",
            "\n",
            "    .colab-df-convert {\n",
            "      background-color: #E8F0FE;\n",
            "      border: none;\n",
            "      border-radius: 50%;\n",
            "      cursor: pointer;\n",
            "      display: none;\n",
            "      fill: #1967D2;\n",
            "      height: 32px;\n",
            "      padding: 0 0 0 0;\n",
            "      width: 32px;\n",
            "    }\n",
            "\n",
            "    .colab-df-convert:hover {\n",
            "      background-color: #E2EBFA;\n",
            "      box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px
1px rgba(60, 64, 67, 0.15);\n",
            "      fill: #174EA6;\n",
            "    }\n",
            "\n",
            "    .colab-df-buttons div {\n",
            "      margin-bottom: 4px;\n",
            "    }\n",
            "\n",
            "    [theme=dark] .colab-df-convert {\n",
            "      background-color: #3B4455;\n",
            "      fill: #D2E3FC;\n",
            "    }\n",
            "\n",
            "    [theme=dark] .colab-df-convert:hover {\n",
            "      background-color: #434B5C;\n",
            "      box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",
            "      filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",
            "      fill: #FFFFFF;\n",
            "    }\n",
```

```
              "  </style>\n",
              "\n",
              "    <script>\n",
              "      const buttonEl =\n",
              "        document.querySelector('#df-d458d352-7bbe-4a6f-85d7-
673ac43ce016 button.colab-df-convert');\n",
              "      buttonEl.style.display =\n",
              "        google.colab.kernel.accessAllowed ? 'block' : 'none';\n",
              "\n",
              "      async function convertToInteractive(key) {\n",
              "        const element = document.querySelector('#df-d458d352-
7bbe-4a6f-85d7-673ac43ce016');\n",
              "        const dataTable =\n",
              "          await
google.colab.kernel.invokeFunction('convertToInteractive',\n",
              "                                                    [key],
{});\n",
              "        if (!dataTable) return;\n",
              "\n",
              "        const docLinkHtml = 'Like what you see? Visit the ' +\n",
              "          '<a target=\"_blank\"
href=https://colab.research.google.com/notebooks/data_table.ipynb>data table
notebook</a>'\n",
              "          + ' to learn more about interactive tables.';\n",
              "        element.innerHTML = '';\n",
              "        dataTable['output_type'] = 'display_data';\n",
              "        await google.colab.output.renderOutput(dataTable,
element);\n",
              "        const docLink = document.createElement('div');\n",
              "        docLink.innerHTML = docLinkHtml;\n",
              "        element.appendChild(docLink);\n",
              "      }\n",
              "    </script>\n",
              "  </div>\n",
              "\n",
              "\n",
              "<div id=\"df-e5cf2fa9-e53c-4a75-9f58-3702bdc1a942\">\n",
              "  <button class=\"colab-df-quickchart\" onclick=\"quickchart('df-
e5cf2fa9-e53c-4a75-9f58-3702bdc1a942')\"\n",
              "            title=\"Suggest charts\"\n",
              "            style=\"display:none;\">\n",
              "\n",
              "<svg xmlns=\"http://www.w3.org/2000/svg\"
height=\"24px\"viewBox=\"0 0 24 24\"\n",
              "     width=\"24px\">\n",
              "    <g>\n",
              "        <path d=\"M19 3H5c-1.1 0-2 .9-2 2v14c0 1.1.9 2 2 2h14c1.1
0 2-.9 2-2V5c0-1.1-.9-2-2-2zM9 17H7v-7h2v7zm4 0h-2V7h2v10zm4 0h-2v-
4h2v4z\"/>\n",
              "    </g>\n",
              "</svg>\n",
              "  </button>\n",
              "\n",
              "<style>\n",
              "  .colab-df-quickchart {\n",
              "      --bg-color: #E8F0FE;\n",
              "      --fill-color: #1967D2;\n",
              "      --hover-bg-color: #E2EBFA;\n",
              "      --hover-fill-color: #174EA6;\n",
              "      --disabled-fill-color: #AAA;\n",
              "      --disabled-bg-color: #DDD;\n",
              "  }\n",
```

```
            "\n",
            "  [theme=dark] .colab-df-quickchart {\n",
            "      --bg-color: #3B4455;\n",
            "      --fill-color: #D2E3FC;\n",
            "      --hover-bg-color: #434B5C;\n",
            "      --hover-fill-color: #FFFFFF;\n",
            "      --disabled-bg-color: #3B4455;\n",
            "      --disabled-fill-color: #666;\n",
            "  }\n",
            "\n",
            "  .colab-df-quickchart {\n",
            "    background-color: var(--bg-color);\n",
            "    border: none;\n",
            "    border-radius: 50%;\n",
            "    cursor: pointer;\n",
            "    display: none;\n",
            "    fill: var(--fill-color);\n",
            "    height: 32px;\n",
            "    padding: 0;\n",
            "    width: 32px;\n",
            "  }\n",
            "\n",
            "  .colab-df-quickchart:hover {\n",
            "    background-color: var(--hover-bg-color);\n",
            "    box-shadow: 0 1px 2px rgba(60, 64, 67, 0.3), 0 1px 3px 1px
rgba(60, 64, 67, 0.15);\n",
            "    fill: var(--button-hover-fill-color);\n",
            "  }\n",
            "\n",
            "  .colab-df-quickchart-complete:disabled,\n",
            "  .colab-df-quickchart-complete:disabled:hover {\n",
            "    background-color: var(--disabled-bg-color);\n",
            "    fill: var(--disabled-fill-color);\n",
            "    box-shadow: none;\n",
            "  }\n",
            "\n",
            "  .colab-df-spinner {\n",
            "    border: 2px solid var(--fill-color);\n",
            "    border-color: transparent;\n",
            "    border-bottom-color: var(--fill-color);\n",
            "    animation:\n",
            "      spin 1s steps(1) infinite;\n",
            "  }\n",
            "\n",
            "  @keyframes spin {\n",
            "    0% {\n",
            "      border-color: transparent;\n",
            "      border-bottom-color: var(--fill-color);\n",
            "      border-left-color: var(--fill-color);\n",
            "    }\n",
            "    20% {\n",
            "      border-color: transparent;\n",
            "      border-left-color: var(--fill-color);\n",
            "      border-top-color: var(--fill-color);\n",
            "    }\n",
            "    30% {\n",
            "      border-color: transparent;\n",
            "      border-left-color: var(--fill-color);\n",
            "      border-top-color: var(--fill-color);\n",
            "      border-right-color: var(--fill-color);\n",
            "    }\n",
            "    40% {\n",
```

```
                "      border-color: transparent;\n",
                "      border-right-color: var(--fill-color);\n",
                "      border-top-color: var(--fill-color);\n",
                "    }\n",
                "    60% {\n",
                "      border-color: transparent;\n",
                "      border-right-color: var(--fill-color);\n",
                "    }\n",
                "    80% {\n",
                "      border-color: transparent;\n",
                "      border-right-color: var(--fill-color);\n",
                "      border-bottom-color: var(--fill-color);\n",
                "    }\n",
                "    90% {\n",
                "      border-color: transparent;\n",
                "      border-bottom-color: var(--fill-color);\n",
                "    }\n",
                "  }\n",
                "</style>\n",
                "\n",
                "  <script>\n",
                "    async function quickchart(key) {\n",
                "      const quickchartButtonEl =\n",
                "        document.querySelector('#' + key + ' button');\n",
                "      quickchartButtonEl.disabled = true;  // To prevent multiple
clicks.\n",
                "      quickchartButtonEl.classList.add('colab-df-spinner');\n",
                "      try {\n",
                "        const charts = await
google.colab.kernel.invokeFunction(\n",
                "            'suggestCharts', [key], {});\n",
                "      } catch (error) {\n",
                "        console.error('Error during call to suggestCharts:',
error);\n",
                "      }\n",
                "      quickchartButtonEl.classList.remove('colab-df-
spinner');\n",
                "      quickchartButtonEl.classList.add('colab-df-quickchart-
complete');\n",
                "    }\n",
                "    (() => {\n",
                "      let quickchartButtonEl =\n",
                "        document.querySelector('#df-e5cf2fa9-e53c-4a75-9f58-
3702bdc1a942 button');\n",
                "      quickchartButtonEl.style.display =\n",
                "        google.colab.kernel.accessAllowed ? 'block' : 'none';\n",
                "    })();\n",
                "  </script>\n",
                "</div>\n",
                "    </div>\n",
                "  </div>\n"
              ]
            },
            "metadata": {},
            "execution_count": 2
          }
        ]
      },
      {
        "cell_type": "code",
        "source": [
          "missing_values = data.isnull().sum()\n",
```

```
      "missing_values"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "568822SAJyhr",
      "outputId": "dfc3442c-d60e-4366-e984-76a8a62a7330"
    },
    "execution_count": 3,
    "outputs": [
      {
        "output_type": "execute_result",
        "data": {
          "text/plain": [
            "Time      0\n",
            "V1        0\n",
            "V2        0\n",
            "V3        0\n",
            "V4        0\n",
            "V5        0\n",
            "V6        0\n",
            "V7        0\n",
            "V8        0\n",
            "V9        0\n",
            "V10       0\n",
            "V11       0\n",
            "V12       0\n",
            "V13       0\n",
            "V14       0\n",
            "V15       0\n",
            "V16       0\n",
            "V17       0\n",
            "V18       0\n",
            "V19       0\n",
            "V20       0\n",
            "V21       0\n",
            "V22       0\n",
            "V23       0\n",
            "V24       0\n",
            "V25       0\n",
            "V26       0\n",
            "V27       0\n",
            "V28       0\n",
            "Amount    0\n",
            "Class     0\n",
            "dtype: int64"
          ]
        },
        "metadata": {},
        "execution_count": 3
      }
    ]
  },
  {
    "cell_type": "code",
    "source": [
      "target_distribution = data['Class'].value_counts()\n",
      "target_distribution"
    ],
    "metadata": {
      "colab": {
```

```
      "base_uri": "https://localhost:8080/"
    },
    "id": "NNjz7PqjJ7Ab",
    "outputId": "b4d29141-1c54-485b-be70-3d499a644add"
  },
  "execution_count": 4,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "0    763\n",
          "1      9\n",
          "Name: Class, dtype: int64"
        ]
      },
      "metadata": {},
      "execution_count": 4
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "X = data.drop('Class', axis=1)\n",
    "y = data['Class']"
  ],
  "metadata": {
    "id": "dwAN_OCTKPD7"
  },
  "execution_count": 5,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "X\n",
    "y"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "HI0AhUvfK7OQ",
    "outputId": "333eeb4c-c2b2-4f4b-8104-08bc52252613"
  },
  "execution_count": 6,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "0      0\n",
          "1      1\n",
          "2      0\n",
          "3      0\n",
          "4      0\n",
          "      ..\n",
          "767    0\n",
          "768    0\n",
          "769    0\n",
          "770    0\n",
```

```
              "771    0\n",
              "Name: Class, Length: 772, dtype: int64"
            ]
          },
          "metadata": {},
          "execution_count": 6
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from sklearn.model_selection import train_test_split\n",
        "\n",
        "X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)"
      ],
      "metadata": {
        "id": "fwfl7L4MOgsK"
      },
      "execution_count": 7,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "from imblearn.over_sampling import SMOTE\n",
        "from imblearn.combine import SMOTEENN\n",
        "\n",
        "smote = SMOTE(random_state=42)\n",
        "X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
y_train)"
      ],
      "metadata": {
        "id": "u1cfK8zFOu5Q"
      },
      "execution_count": 8,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "from sklearn.linear_model import LogisticRegression\n",
        "from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier\n",
        "from sklearn.svm import SVC\n",
        "\n",
        "models = {\n",
        "    'M1': LogisticRegression(),\n",
        "    'M2': RandomForestClassifier(),\n",
        "    'M3': GradientBoostingClassifier(),\n",
        "    'M4': SVC(),\n",
        "}\n"
      ],
      "metadata": {
        "id": "saN4ux95O-5n"
      },
      "execution_count": 16,
      "outputs": []
    },
    {
      "cell_type": "code",
```

```
      "source": [
        "from sklearn.metrics import accuracy_score\n",
        "\n",
        "results = {}\n",
        "for model_name, model in models.items():\n",
        "    model.fit(X_train, y_train)\n",
        "    y_pred = model.predict(X_test)\n",
        "    original_accuracy = accuracy_score(y_test, y_pred)\n",
        "\n",
        "    model.fit(X_train_resampled, y_train_resampled)\n",
        "    y_pred_resampled = model.predict(X_test)\n",
        "    resampled_accuracy = accuracy_score(y_test, y_pred_resampled)\n",
        "\n",
        "    results[model_name] = {'Original Accuracy': original_accuracy,
'Resampled Accuracy': resampled_accuracy}\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "qdlMs2NKRGo2",
        "outputId": "0a5ddf31-fd9c-4482-da25-c5948d5cd0e2"
      },
      "execution_count": 17,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):\n",
            "STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",
            "\n",
            "Increase the number of iterations (max_iter) or scale the data as
shown in:\n",
            "    https://scikit-learn.org/stable/modules/preprocessing.html\n",
            "Please also refer to the documentation for alternative solver
options:\n",
            "    https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression\n",
            "  n_iter_i = _check_optimize_result(\n",
            "/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):\n",
            "STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",
            "\n",
            "Increase the number of iterations (max_iter) or scale the data as
shown in:\n",
            "    https://scikit-learn.org/stable/modules/preprocessing.html\n",
            "Please also refer to the documentation for alternative solver
options:\n",
            "    https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression\n",
            "  n_iter_i = _check_optimize_result(\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
```

```
        "print(\"\\nResults:\")\n",
        "for model_name, result in results.items():\n",
        "    print(f\"Model: {model_name}\")\n",
        "    print(f\"Original Accuracy: {result['Original
Accuracy']:.4f}\")\n",
        "    print(f\"Resampled Accuracy: {result['Resampled
Accuracy']:.4f}\")\n",
        "    print()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "s5fbrP3QRRFl",
        "outputId": "72bce6bc-adf7-4a18-ef1a-90a0380c0c1b"
      },
      "execution_count": 18,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "\n",
            "Results:\n",
            "Model: M1\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.8774\n",
            "\n",
            "Model: M2\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.9935\n",
            "\n",
            "Model: M3\n",
            "Original Accuracy: 0.9871\n",
            "Resampled Accuracy: 0.9935\n",
            "\n",
            "Model: M4\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.6710\n",
            "\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from imblearn.over_sampling import RandomOverSampler\n",
        "\n",
        "ros = RandomOverSampler(random_state=42)\n",
        "X_train_resampled_1, y_train_resampled_1 = ros.fit_resample(X_train,
y_train)"
      ],
      "metadata": {
        "id": "ewVhU4rWTqPM"
      },
      "execution_count": 22,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
```

```
        "from sklearn.metrics import accuracy_score\n",
        "\n",
        "results = {}\n",
        "for model_name, model in models.items():\n",
        "    model.fit(X_train, y_train)\n",
        "    y_pred = model.predict(X_test)\n",
        "    original_accuracy = accuracy_score(y_test, y_pred)\n",
        "\n",
        "    model.fit(X_train_resampled_1, y_train_resampled_1)\n",
        "    y_pred_resampled_1 = model.predict(X_test)\n",
        "    resampled_accuracy = accuracy_score(y_test, y_pred_resampled_1)\n",
        "\n",
        "    results[model_name] = {'Original Accuracy': original_accuracy,
'Resampled Accuracy': resampled_accuracy}\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "0oNLLK6AT3Em",
        "outputId": "7f9eccfe-b161-4c73-f27e-cb55bdeb807e"
      },
      "execution_count": 23,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):\n",
            "STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",
            "\n",
            "Increase the number of iterations (max_iter) or scale the data as
shown in:\n",
            "    https://scikit-learn.org/stable/modules/preprocessing.html\n",
            "Please also refer to the documentation for alternative solver
options:\n",
            "    https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression\n",
            "  n_iter_i = _check_optimize_result(\n",
            "/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):\n",
            "STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",
            "\n",
            "Increase the number of iterations (max_iter) or scale the data as
shown in:\n",
            "    https://scikit-learn.org/stable/modules/preprocessing.html\n",
            "Please also refer to the documentation for alternative solver
options:\n",
            "    https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression\n",
            "  n_iter_i = _check_optimize_result(\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "print(\"\\nResults:\")\n",
```

```
        "for model_name, result in results.items():\n",
        "    print(f\"Model: {model_name}\")\n",
        "    print(f\"Original Accuracy: {result['Original
Accuracy']:.4f}\")\n",
        "    print(f\"Resampled Accuracy: {result['Resampled
Accuracy']:.4f}\")\n",
        "    print()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "UtQUvVBCUIlu",
        "outputId": "c2fa2b8b-a1b4-438c-f606-dccea9a97f16"
      },
      "execution_count": 24,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "\n",
            "Results:\n",
            "Model: M1\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.8774\n",
            "\n",
            "Model: M2\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.9935\n",
            "\n",
            "Model: M3\n",
            "Original Accuracy: 0.9871\n",
            "Resampled Accuracy: 0.9935\n",
            "\n",
            "Model: M4\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.6968\n",
            "\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from imblearn.under_sampling import NearMiss\n",
        "\n",
        "nm = NearMiss()\n",
        "X_train_resampled_2, y_train_resampled_2 = nm.fit_resample(X_train,
y_train)\n",
        "results = {}\n",
        "for model_name, model in models.items():\n",
        "    model.fit(X_train, y_train)\n",
        "    y_pred = model.predict(X_test)\n",
        "    original_accuracy = accuracy_score(y_test, y_pred)\n",
        "\n",
        "    model.fit(X_train_resampled_2, y_train_resampled_2)\n",
        "    y_pred_resampled_2 = model.predict(X_test)\n",
        "    resampled_accuracy = accuracy_score(y_test, y_pred_resampled_2)\n",
        "\n",
        "    results[model_name] = {'Original Accuracy': original_accuracy,
```

```
'Resampled Accuracy': resampled_accuracy}\n",
        "\n",
        "print(\"\\nResults:\")\n",
        "for model_name, result in results.items():\n",
        "    print(f\"Model: {model_name}\")\n",
        "    print(f\"Original Accuracy: {result['Original
Accuracy']:.4f}\")\n",
        "    print(f\"Resampled Accuracy: {result['Resampled
Accuracy']:.4f}\")\n",
        "    print()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "lB-Sg5CYWT7v",
        "outputId": "a5081768-baa7-42ee-9b54-5892cb3a55a2"
      },
      "execution_count": 25,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):\n",
            "STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",
            "\n",
            "Increase the number of iterations (max_iter) or scale the data as
shown in:\n",
            "    https://scikit-learn.org/stable/modules/preprocessing.html\n",
            "Please also refer to the documentation for alternative solver
options:\n",
            "    https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression\n",
            "  n_iter_i = _check_optimize_result(\n",
            "/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):\n",
            "STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",
            "\n",
            "Increase the number of iterations (max_iter) or scale the data as
shown in:\n",
            "    https://scikit-learn.org/stable/modules/preprocessing.html\n",
            "Please also refer to the documentation for alternative solver
options:\n",
            "    https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression\n",
            "  n_iter_i = _check_optimize_result(\n"
          ]
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "\n",
            "Results:\n",
            "Model: M1\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.4323\n",
            "\n",
```

```
            "Model: M2\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.5742\n",
            "\n",
            "Model: M3\n",
            "Original Accuracy: 0.9871\n",
            "Resampled Accuracy: 0.0645\n",
            "\n",
            "Model: M4\n",
            "Original Accuracy: 0.9935\n",
            "Resampled Accuracy: 0.3484\n",
            "\n"
          ]
        }
      ]
    }
  ]
}
```