higgs-ml-project/notebooks/final_report.ipynb

{ "cells": [{ "cell_type": "markdown", "metadata": {}, "source": ["# Machine Learning Pipeline: Feature Selection and Hyperparameter Optimization\n", "## Üsküdar Üniversitesi Fen Bilimleri Enstitüsü\n", "### Makine Öğrenmesi Final Ödevi"] }, { "cell_type": "markdown", "metadata": {}, "source": ["## 1. Giriş ve Veri Seti Tanıtımı"] }, { "cell_type": "code", "execution_count": null, "metadata": {}, "outputs": [], "source": ["# Import libraries\n", "import numpy as np\n", "import pandas as pd\n", "import matplotlib.pyplot as plt\n", "import seaborn as sns\n", "from sklearn.model_selection import train_test_split\n", "from pathlib import Path\n", "\n", "# Set global styles\n", "plt.style.use('ggplot')\n", "sns.set_palette('viridis')\n", "pd.set_option('display.float_format', '{:.4f}'.format)"] }, { "cell_type": "code", "execution_count": null, "metadata": {}, "outputs": [], "source": ["# Load the data\n", "data_path = Path('../data/higgs_sample.csv')\n", "df = pd.read_csv(data_path, header=None)\n", "\n", "# Name columns according to HIGGS dataset documentation\n", "columns = ['class_label'] + [f'feature_{i}' for i in range(1, 29)]\n", "df.columns = columns\n", "\n", "# Display dataset info\n", "print(f"Dataset shape: {df.shape}")\n", "print("\nFirst 5 rows:")\n", "display(df.head())\n", "\n", "# Class distribution\n", "class_dist = df['class_label'].value_counts(normalize=True)\n", "print("\nClass distribution:")\n", "display(class_dist)"] }, { "cell_type": "markdown", "metadata": {}, "source": ["## 2. Veri Ön İşleme\n", "### Aykırı Değer Analizi ve Ölçekleme"] }, { "cell_type": "code", "execution_count": null, "metadata": {}, "outputs": [], "source": ["from src.preprocessing import OutlierCapper\n", "from sklearn.preprocessing import MinMaxScaler\n", "\n", "# Separate features and target\n", "X = df.drop('class_label', axis=1)\n", "y = df['class_label']\n", "\n", "# Outlier handling\n", "capper = OutlierCapper(factor=1.5)\n", "X_capped = pd.DataFrame(capper.fit_transform(X), columns=X.columns)\n", "\n", "# Visualize before/after for sample features\n", "fig, axes = plt.subplots(2, 2, figsize=(12, 8))\n", "sns.boxplot(data=X[['feature_1', 'feature_2']], ax=axes[0, 0])\n", "axes[0, 0].set_title('Original Features (Sample)')\n", "\n", "sns.boxplot(data=X_capped[['feature_1', 'feature_2']], ax=axes[0, 1])\n", "axes[0, 1].set_title('After Outlier Capping (Sample)')\n", "\n", "# Scaling\n", "scaler = MinMaxScaler()\n", "X_scaled = pd.DataFrame(scaler.fit_transform(X_capped), columns=X.columns)\n", "\n", "# Visualize scaling results\n", "sns.histplot(X_scaled['feature_1'], kde=True, ax=axes[1, 0])\n", "axes[1, 0].set_title('Scaled Feature 1 Distribution')\n", "\n", "sns.histplot(X_scaled['feature_2'], kde=True, ax=axes[1, 1])\n", "axes[1, 1].set_title('Scaled Feature 2 Distribution')\n", "\n", "plt.tight_layout()\n", "plt.savefig('../outputs/figures/preprocessing_results.png', dpi=300)\n", "plt.show()"] }, { "cell_type": "markdown", "metadata": {}, "source": ["## 3. Özellik Seçimi\n", "### ANOVA F-Skor ile En Önemli 15 Özelliğin Seçimi"] }, { "cell_type": "code", "execution_count": null, "metadata": {}, "outputs": [], "source": ["from src.feature_selection import select_features\n", "from sklearn.feature_selection import f_classif\n", "\n", "# Feature selection\n", "X_selected, selected_idx = select_features(X_scaled.values, y.values, method='anova', k=15)\n", "\n", "# Get selected feature names\n", "selected_features = X_scaled.columns[selected_idx].tolist()\n", "print(f"Selected {len(selected_features)} features:")\n", "print(selected_features)\n", "\n", "# Calculate ANOVA F-scores for all features\n", "f_scores, _ = f_classif(X_scaled, y)\n", "feature_scores = pd.DataFrame({\n", " 'Feature': X_scaled.columns,\n", " 'F_Score': f_scores\n", "}).sort_values('F_Score', ascending=False)\n", "\n", "# Visualize feature importance\n", "plt.figure(figsize=(12, 8))\n", "sns.barplot(x='F_Score', y='Feature', data=feature_scores.head(20))\n", "plt.title('Top 20 Features by ANOVA F-Score')\n", "plt.xlabel('F-Score')\n", "plt.ylabel('Feature')\n", "plt.tight_layout()\n", "plt.savefig('../outputs/figures/feature_importance.png', dpi=300)\n", "plt.show()"] }, { "cell_type": "markdown", "metadata": {}, "source": ["## 4. Modelleme ve Değerlendirme\n", "### İç İçe Çapraz Doğrulama (Nested Cross-Validation) ile Model Performansı"] }, { "cell_type": "code", "execution_count": null, "metadata": {}, "outputs": [], "source": ["import joblib\n", "from src.modeling import get_models\n", "from src.evaluation import evaluate_model\n", "\n", "# Load pre-trained models and results\n", "model_results = {}\n", "models = get_models()\n", "\n", "for model_name in models.keys():\n", " try:\n", " # Load metrics\n", " metrics_path = f'../outputs/results/{model_name}metrics.csv'\n", " metrics_df = pd.read_csv(metrics_path)\n", " \n", " # Load best model from first fold\n", " model_path = f'../outputs/models/{model_name}_fold1.pkl'\n", " model = joblib.load(model_path)\n", " \n", " model_results[model_name] = {\n", " 'metrics': metrics_df,\n", " 'model': model\n", " }\n", " print(f"Loaded results for {model_name}")\n", " except FileNotFoundError:\n", " print(f"Results not found for {model_name}")\n", "\n", "# Calculate average metrics\n", "performance_summary = []\n", "for model_name, data in model_results.items():\n", " avg_metrics = data['metrics'].mean().to_dict()\n", " avg_metrics['Model'] = model_name\n", " performance_summary.append(avg_metrics)\n", "\n", "performance_df = pd.DataFrame(performance_summary)\n", "performance_df = performance_df[['Model', 'Accuracy', 'Precision', 'Recall', 'F1', 'ROC-AUC']]\n", "print("\nAverage Performance Metrics:")\n", "display(performance_df)"] }, { "cell_type": "markdown", "metadata": {}, "source": ["## 5. Performans Karşılaştırması ve ROC Eğrileri"] }, { "cell_type": "code", "execution_count": null, "metadata": {}, "outputs": [], "source": ["# Prepare test set for final evaluation\n", "X_train, X_test, y_train, y_test = train_test_split(\n", " X_scaled[selected_features], y, test_size=0.2, random_state=42, stratify=y\n", ")\n", "\n", "# Plot ROC curves for all models\n", "plt.figure(figsize=(10, 8))\n", "\n", "for model_name, data in model_results.items():\n", " model = data['model']\n", " \n", " if hasattr(model, "predict_proba"):\n", " y_proba = model.predict_proba(X_test)[:, 1]\n", " else: # For SVM without probability\n", " decision = model.decision_function(X_test)\n", " y_proba = (decision - decision.min()) / (decision.max() - decision.min())\n", " \n", " fpr, tpr, _ = roc_curve(y_test, y_proba)\n", " roc_auc = auc(fpr, tpr)\n", " \n", " plt.plot(fpr, tpr, lw=2, label=f'{model_name} (AUC = {roc_auc:.3f})')\n", "\n", "plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')\n", "plt.xlim([0.0, 1.0])\n", "plt.ylim([0.0, 1.05])\n", "plt.xlabel('False Positive Rate')\n", "plt.ylabel('True Positive Rate')\n", "plt.title('ROC Curve Comparison')\n", "plt.legend(loc="lower right")\n", "plt.grid(True)\n", "plt.savefig('../outputs/figures/roc_comparison.png', dpi=300)\n", "plt.show()"] }, { "cell_type": "markdown", "metadata": {}, "source": ["## 6. Sonuçlar ve Yorum\n", "### En Başarılı Model Analizi"] }, { "cell_type": "code", "execution_count": null, "metadata": {}, "outputs": [], "source": ["# Identify best model\n", "best_model_name = performance_df.sort_values('ROC-AUC', ascending=False).iloc[0]['Model']\n", "best_model = model_results[best_model_name]['model']\n", "best_metrics = performance_df[performance_df['Model'] == best_model_name].iloc[0]\n", "\n", "print(f"Best Performing Model:

performance_df[performance_df['Model'] == best_model_name].iloc[0]\n", "\n", "print(f"Best Performing Model: {best_model_name}")\n", "print(f"ROC-AUC: {best_metrics['ROC-AUC']:.4f}")\n", "print(f"Accuracy: {best_metrics['Accuracy']:.4f}")\n", "print(f"F1 Score: {best_metrics['F1']:.4f}")\n", "\n", "# Show best hyperparameters\n", "if hasattr(best_model, 'best_params'):\n", " print("\nBest Hyperparameters:")\n", " display(pd.Series(best_model.best_params_).to_frame('Value'))"] }, { "cell_type": "markdown", "metadata": {}, "source": ["## 7. Proje Özeti ve Değerlendirme"] }], "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": "3.11.5" } }, "nbformat": 4, "nbformat_minor": 4 }