Redukce dimenze dat

Václav Honzík
A21N0045P

Zadání

Pro vhodně zvolená vícerozměrná data s alespoň jednou kategoriální proměnnou proveďte redukci dimenze dat. Následně proveďte diskriminační analýzu na 1D nebo 2D redukovaná data. Kvalitu diskriminace posuďte na základě cross-validation procesu.

Zvolený dataset

Jako dataset, pro který budu dimenzionalitu redukovat, jsem se rozhodl vybrat **Wine dataset**, který je k dispozici zde: https://archive.ics.uci.edu/ml/datasets/wine. Jedná se o dataset vín, který obsahuje 1 kategoriální proměnnou - typ vína a 13 vlastností, jako např. množství alkoholu, barevnou intenzitu, množství horčíku, apod.

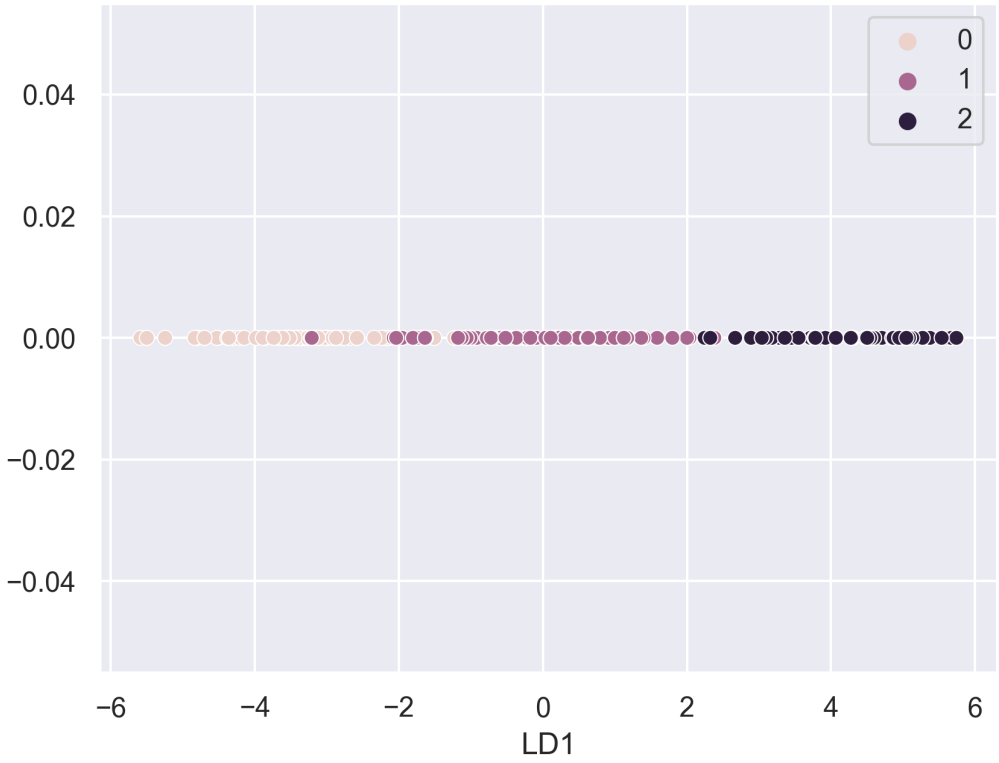
Program

Program je napsaný v programovacím jazyce Python s použití několika matematických a vizualizačních knihoven - NumPy, Pandas, Seaborn a Sklearn. Kód je napsaný formou interaktivního Jupyter Notebooku.

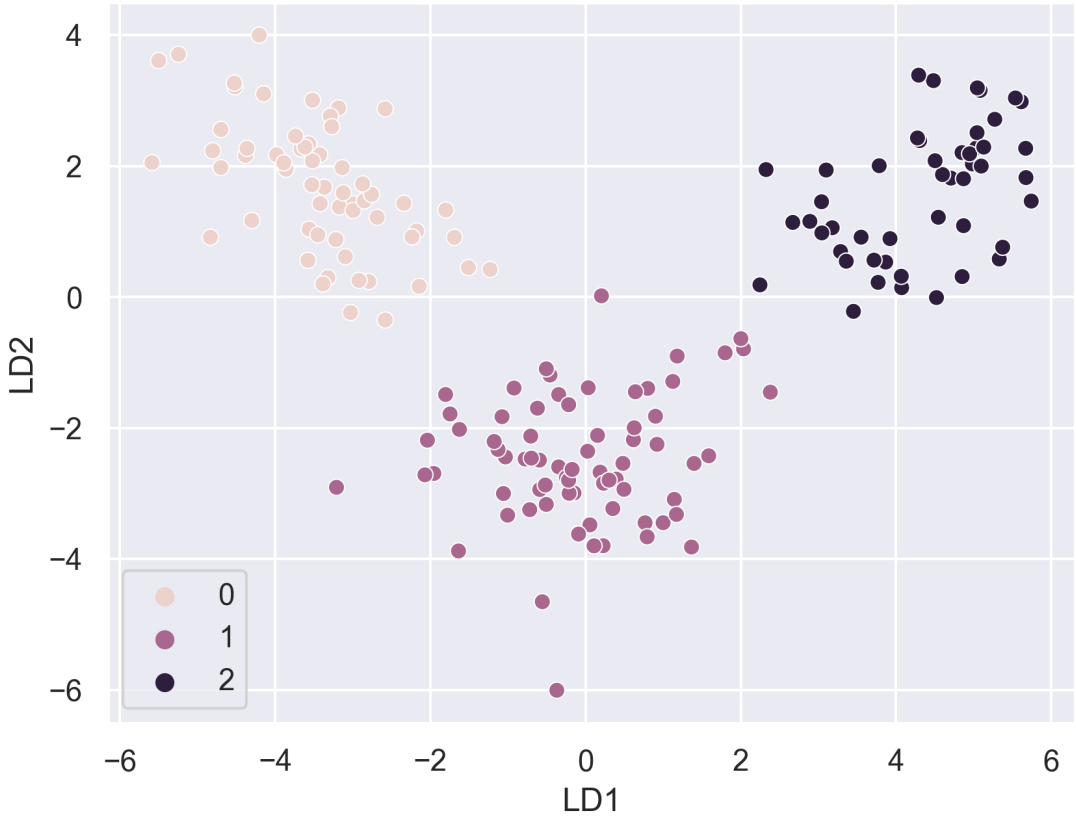
Redukce dimenzionality

Pro redukci dimenzionality jsem použil metodu LDA (Linear Discriminant Analysis), která je zobecněním Fisherovo lineární diskriminace. Výpočet je provedený pomocí SVD dekompozice v knihovně scikit / sklearn.

Po provedení redukce dimenzionality dostaneme transformované 1D / 2D data, které můžeme snadno vizualizovat. Vizualizace v 1D je vidět na Obr. 1, zatímco Obr. 2 obsahuje data ve dvou dimenzích



Obrázek 1 - Vizualizace dat v 1D



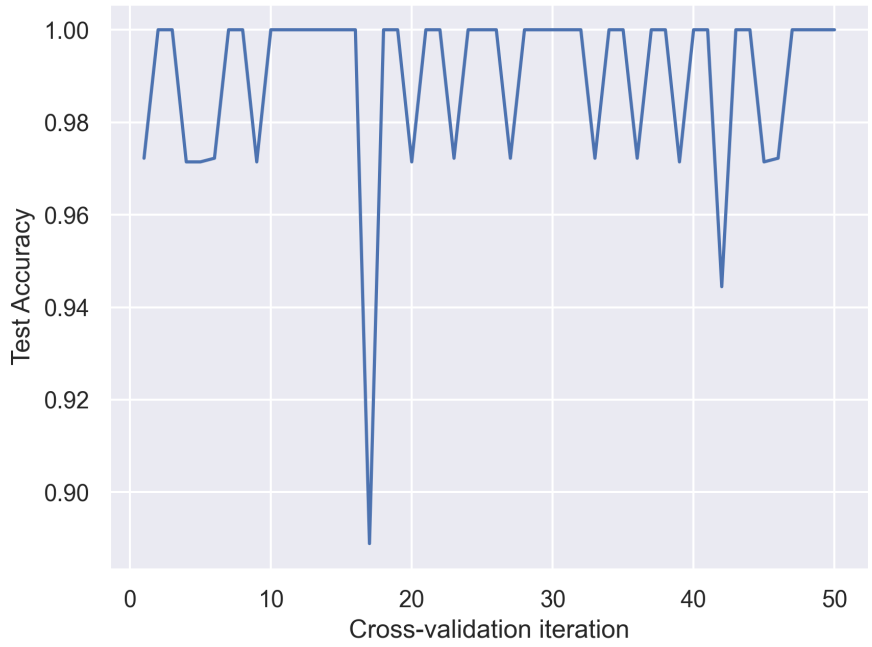
Obrázek 2 - Vizualizace ve 2D

Jak můžeme vidět, pro tento dataset funguje LDA velmi dobře a pro výsledná data (ve 2D) můžeme vytvořit klasifikátor pouze pomocí dvou přímek. S horší přesností by bylo možné i použít i pouze jednu dimenzi.

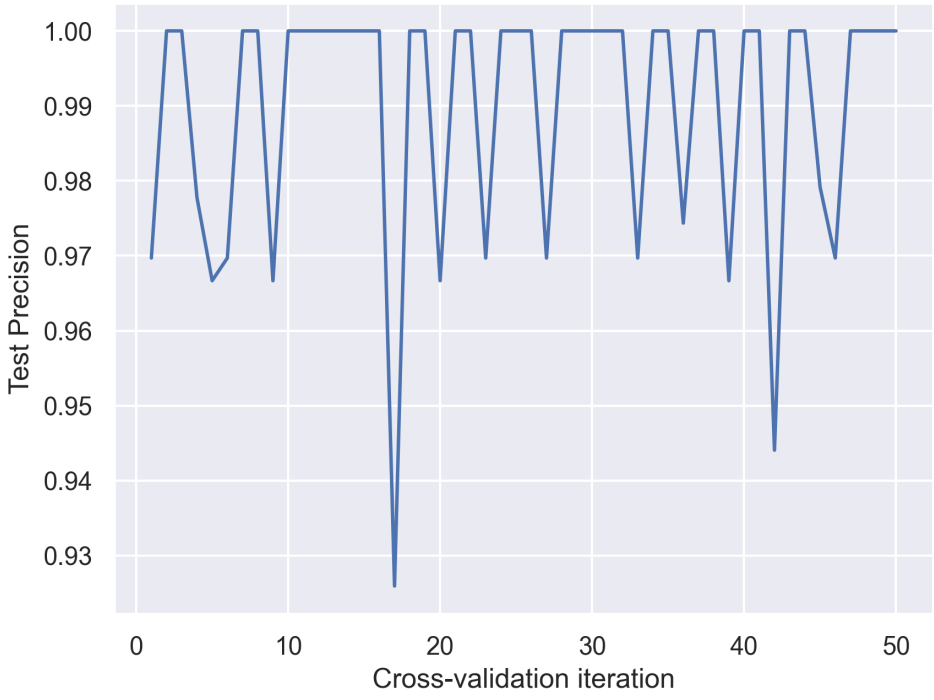
Cross-validace

Pro otestování vhodnosti a kvality LDA jsem použil Stratified K-Fold cross validaci, kdy data rozdělíme do K částí (zde jsem zvolil K=5, tzn. 80-20 rozdělení pro trénovací a validační data) a jedna část se použije jako validační data. “Stratified” nám zaručí, že vygenerované části budou mít rovnoměrné rozdělení pro jednotlivé třídy.

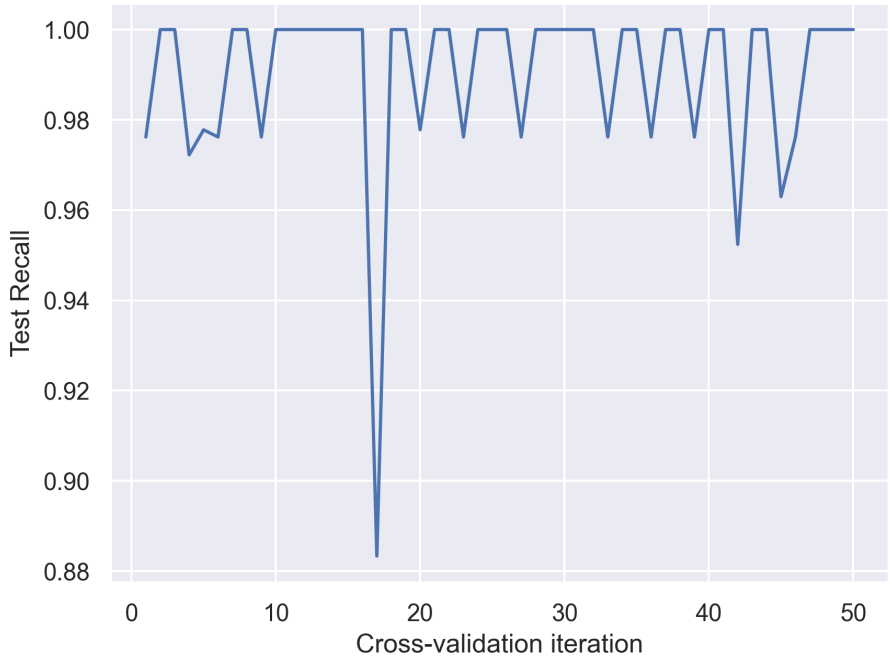
Protože 5-fold cross-validace by nám vrátila pouze 5 vzorků, opakoval jsem celý proces desetkrát. V každé iteraci natrénujeme model na testovacích datech a necháme ho klasifikovat validační data. Pro změření úspěšnosti jsem použil metriky přesnost, precision, recall a F1 skóre. Hodnoty těchto metrik jsou na obrázcích 3, 4, 5 a 6.



Obrázek 3 - Přesnost na validačních datech v každé iteraci



Obrázek 4 - Precision na validačních datech v každé iteraci



Obrázek 5 - Recall na validačních datech v každé iteraci

Obrázek 6 - F1 skóre na validačních datech v každé iteraci

V mnoha případech dokáže model mít až 100% přesnost (tím pádem i precision, recall a F1 skóre), což ale není překvapivé, protože jsou data velmi snadno oddělitelná, jak již bylo vidět na Obr. 2.

V průměru jsem dostal přesnost na validačních datech 98.9%, precision 99%, recall 99% a F1 98.9%.

Spuštění

Pro spuštění je potřeba mít nainstalované všechny závislosti, které jsou specifikované v souboru *requirements.txt*. Závislosti lze nainstalovat pomocí příkazu:

```
pip install -r requirements.txt
```

Alternativně lze nainstalovat nejnovější verze přímo pomocí příkazu:

```
pip install pandas matplotlib scipy seaborn jupyter notebook numpy
```

Pro spuštění notebooku je nutné otevřít příkazovou řádku v kořenovém adresáři projektu a zadat příkaz:

```
jupyter notebook
```

Který na adrese http://localhost:8888/ spustí webovou aplikaci Jupyter. Následně stačí otevřít notebook **main.ipynb**, který je už možné procházet.

Všechny spuštěné vizualizace se automaticky ukládají do složky *outputs*.