SPRAWOZDANIE

Zajęcia: Uczenie Maszynowe

Prowadzący: prof. dr hab. inż. Vasyl Martsenyuk

Laboratorium Nr 1 Data 27.09.2025 Temat: " Praktyczne zastosowanie regresji liniowej w analizie danych. Implementacja algorytmów klasyfikacji binarnej w Pythonie" Wariant 9	Artur Rolak Informatyka II stopień, stacjonarne, 1semestr, gr.1bS

1. Polecenie: wariant 1 zadania

Opracować przepływ pracy uczenia maszynowego dla zagadnienia regresji (model regresji liniowej) oraz klasyfikacji binarnej (model SVM) na podstawie zbioru danych, zgodnie z wariantem zadania nr 9.

2. Opis programu opracowanego (kody źródłowe, rzuty ekranu)

Celem programu jest zbudowanie dwóch modeli uczenia maszynowego na tym samym zbiorze danych pd_speech_features.csv, zawierającym cechy akustyczne mowy pacjentów z chorobą Parkinsona. Dane pochodzą z badań klinicznych i zawierają zestaw numerycznych współczynników opisujących nagrania głosu. Zadanie polega na przeprowadzeniu dwóch analiz:

1. **Klasyfikacja (SVM)** – Przewidzenie czy pacjent cierpi na chorobę Parkinsona, na podstawie kolumny klasy (class lub status).
2. **Regresja (Linear Regression)** – przewidzenie wartości jednej z ciągłych zmiennych (np. age lub innej wybranej cechy) w oparciu o pozostałe kolumny.

Program został przygotowany w języku **Python** z użyciem bibliotek **pandas**, **scikit-learn** oraz **matplotlib**, w formie interaktywnego notebooka Jupyter. Składa się z następujących etapów:

1. **Wczytanie danych** – plik CSV z danymi mowy pacjentów jest wczytywany do obiektu DataFrame.
2. **Wstępne przetwarzanie** – usuwane są kolumny o stałych wartościach, identyfikatory oraz ewentualne braki danych. Wszystkie cechy numeryczne są standaryzowane (StandardScaler).

3. **Podział danych** – zbiór jest dzielony na część treningową (80%) i testową (20%). Dla klasyfikacji stosowana jest stratyfikacja względem klasy.
4. **Model klasyfikacyjny SVM** – utworzony zostaje potok (Pipeline) zawierający skalowanie i klasyfikator SVC. Następnie wykonywane jest strojenie hiperparametrów (C, kernel, gamma) metodą walidacji krzyżowej (GridSearchCV).
5. **Ocena klasyfikatora** – obliczane są metryki jakości: Accuracy, Precision, Recall, F1 oraz ROC-AUC. Dodatkowo generowana jest macierz pomyłek i wykres krzywej ROC.
6. **Model regresyjny (Linear Regression)** – drugi potok wykorzystuje klasyczny model regresji liniowej do przewidywania wybranej zmiennej ciągłej. Wyniki są oceniane na podstawie błędów MAE, RMSE oraz współczynnika determinacji R^2.
7. **Walidacja i porównanie wyników** – dla obu modeli wykonywana jest 5-krotna walidacja krzyżowa, a uzyskane wyniki są porównywane w kontekście skuteczności i stabilności.

Program pozwala w przejrzysty sposób zrealizować cały cykl uczenia maszynowego: od przygotowania danych, przez modelowanie, aż po ewaluację wyników. Dodatkowo jego struktura jest zgodna z przepływem pracy w środowisku KNIME, dzięki czemu te same kroki można łatwo odwzorować w graficznym workflow.

Zajęcia ML — Wariant 9: Parkinson (`pd_speech_features.csv`)

```
import pandas as pd
import numpy as np
from pathlib import Path

from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold, KFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
                             confusion_matrix, classification_report, RocCurveDisplay)
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

pd.set_option("display.max_columns", 120)
```

```
# === 1. Wczytanie danych
csv_path = Path(r"C:\Users\Administrator\Desktop\Studia\Semestr 1\Uczenie Maszynowe L\Laboratorium 1\pd_speech_features.csv")
assert csv_path.exists(), "Brak pliku pd_speech_features.csv w katalogu notebooka."
df = pd.read_csv(csv_path)

print("Wymiary:", df.shape)
display(df.head(3))
print("Podgląd kolumn:", list(df.columns)[:25], "...")
```

```
Wymiary: (756, 755)
```

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	locAbsJitter	rapJitter	ppq5Jitter	ddpJitter	locShimmer	locDbShimmer	apq3Shimmer	apq5Shimmer	apq11Shimmer	ddaShin
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	0.000018	0.00067	0.00129	0.00200	0.05883	0.517	0.03011	0.03496	0.04828	0.0
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	0.000016	0.00052	0.00112	0.00157	0.05516	0.502	0.02320	0.03675	0.06195	0.0
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	0.000015	0.00057	0.00111	0.00171	0.09902	0.897	0.05094	0.06497	0.07772	0.1

3 rows × 755 columns

```
Podgląd kolumn: ['id', 'gender', 'PPE', 'DFA', 'RPDE', 'numPulses', 'numPeriodsPulses', 'meanPeriodPulses', 'stdDevPeriodPulses', 'locPctJitter', 'locAbsJitter', 'rapJitter', 'ppq5Jitter', 'ddpJitter', 'locShimmer', 'locDbShimmer',
'apq3Shimmer', 'apq5Shimmer', 'apq11Shimmer', 'ddaShimmer', 'meanAutoCorrHarmonicity', 'meanNoiseToHarmHarmonicity', 'meanHarmToNoiseHarmonicity', 'minIntensity', 'maxIntensity'] ...
```

[6]:
```
# === 2. Kolumna klasy (binarny cel klasyfikacji)
candidate_labels = [c for c in df.columns if c.lower() in {"class","status","target","label"}]
assert len(candidate_labels) >= 1, "Nie znaleziono kolumny klasy (sprawdź nazwy w pliku)."
LABEL_COL = candidate_labels[0]
print("Kolumna klasy:", LABEL_COL)

# Rzutowanie na int (w razie potrzeby)
df[LABEL_COL] = df[LABEL_COL].astype(int)
print(df[LABEL_COL].value_counts(dropna=False))
```

```
Kolumna klasy: class
class
1    564
0    192
Name: count, dtype: int64
```

[7]:
```
# === 3. Wstępne czyszczenie danych
# Usuwamy kolumny stałe i ewidentne ID (zawierające 'id' lub 'name')
drop_cols = []
for col in df.columns:
    if df[col].nunique(dropna=True) <= 1:
        drop_cols.append(col)
    if "id" in col.lower() or "name" in col.lower():
        drop_cols.append(col)

drop_cols = sorted(set(drop_cols) - {LABEL_COL})
if drop_cols:
    print("Usuwam stałe/ID:", drop_cols)
    df = df.drop(columns=drop_cols)

# Obsługa braków danych — startowo: usuń wiersze z brakami (prosto i bezpiecznie)
before = df.shape
df = df.dropna(axis=0)
after = df.shape
print(f"Po dropna: {before} -> {after}")
```

```
Usuwam stałe/ID: ['id']
Po dropna: (756, 754) -> (756, 754)
```

[8]:
```
# === 4. Regresja: wybór zmiennej ciągłej
# Możesz wymusić konkretny cel, np.: REGRESSION_TARGET = "age"
REGRESSION_TARGET = None  # <- ustaw własną nazwę kolumny jeśli chcesz nadpisać wybór

if REGRESSION_TARGET is None:
    preferred_targets = ["age"]
    for t in preferred_targets:
        if t in df.columns and t != LABEL_COL:
            REGRESSION_TARGET = t
            break

if REGRESSION_TARGET is None:
    numeric_cols = [c for c in df.select_dtypes(include=[np.number]).columns if c != LABEL_COL]
    variances = df[numeric_cols].var().sort_values(ascending=False)
    REGRESSION_TARGET = variances.index[0]

print("Cel regresji:", REGRESSION_TARGET)
assert REGRESSION_TARGET in df.columns, "Nie znaleziono wybranej kolumny regresji w danych."
assert REGRESSION_TARGET != LABEL_COL, "Cel regresji nie może być kolumną klasy."
```

```
Cel regresji: app_entropy_shannon_10_coef
```

[9]:
```
# === 5. Podział danych
# Klasyfikacja (stratyfikacja)
X_cls = df.drop(columns=[LABEL_COL, REGRESSION_TARGET]) if REGRESSION_TARGET in df.columns else df.drop(columns=[LABEL_COL])
y_cls = df[LABEL_COL]

Xtr_c, Xte_c, ytr_c, yte_c = train_test_split(X_cls, y_cls, test_size=0.2, random_state=42, stratify=y_cls)

# Regresja
X_reg = df.drop(columns=[LABEL_COL, REGRESSION_TARGET])
y_reg = df[REGRESSION_TARGET]

Xtr_r, Xte_r, ytr_r, yte_r = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)

print("Cechy (klasyfikacja)  :", Xtr_c.shape, " / Test:", Xte_c.shape)
print("Cechy (regresja)      :", Xtr_r.shape, " / Test:", Xte_r.shape)
```

```
Cechy (klasyfikacja)  : (604, 752)  / Test: (152, 752)
Cechy (regresja)      : (604, 752)  / Test: (152, 752)
```

[10]:

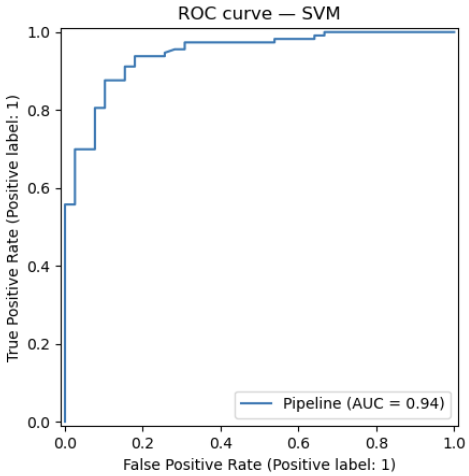
```
# === 6. Pipeline'y + SVM (GridSearch)
num_cols_c = Xtr_c.columns.tolist()
num_cols_r = Xtr_r.columns.tolist()

num_tf_c = ColumnTransformer([("scale", StandardScaler(), num_cols_c)], remainder="drop")
num_tf_r = ColumnTransformer([("scale", StandardScaler(), num_cols_r)], remainder="drop")

svc = SVC(probability=True, class_weight="balanced", random_state=42)
pipe_svc = Pipeline(steps=[("prep", num_tf_c), ("clf", svc)])

param_grid = {
    "clf__kernel": ["linear", "rbf"],
    "clf__C": [0.1, 1, 10, 100],
    "clf__gamma": ["scale", "auto"]  # ignorowane przy kernel='linear'
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
gs = GridSearchCV(pipe_svc, param_grid, cv=cv, scoring="f1", n_jobs=-1)
gs.fit(Xtr_c, ytr_c)

print("=== Najlepszy SVM ===")
print(gs.best_params_)
best_svc = gs.best_estimator_
```

```
=== Najlepszy SVM ===
{'clf__C': 10, 'clf__gamma': 'scale', 'clf__kernel': 'rbf'}
```

[11]:

```
# === 6b. Ewaluacja klasyfikacji
proba = best_svc.predict_proba(Xte_c)[:, 1]
pred = (proba >= 0.5).astype(int)

print("\n--- Klasyfikacja (test) ---")
print("Accuracy :", accuracy_score(yte_c, pred))
print("Precision:", precision_score(yte_c, pred, zero_division=0))
print("Recall   :", recall_score(yte_c, pred))
print("F1       :", f1_score(yte_c, pred))
try:
    print("ROC-AUC  :", roc_auc_score(yte_c, proba))
except Exception as e:
    print("ROC-AUC  (brak):", e)
print("\nRaport:\n", classification_report(yte_c, pred, digits=3))
print("Macierz pomyłek:\n", confusion_matrix(yte_c, pred))

_ = RocCurveDisplay.from_estimator(best_svc, Xte_c, yte_c)
plt.title("ROC curve – SVM")
plt.show()
```

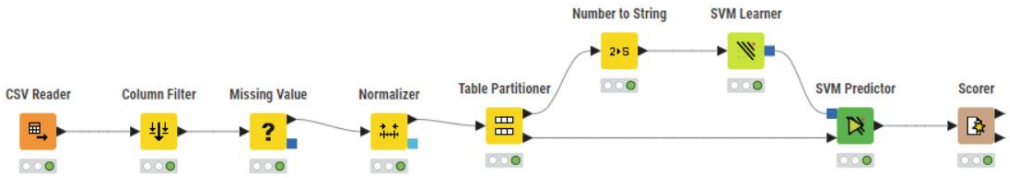
```
--- Klasyfikacja (test) ---
Accuracy : 0.8947368421052632
Precision: 0.907563025210084
Recall   : 0.9557522123893806
F1       : 0.9310344827586207
ROC-AUC  : 0.9442931699568867

Raport:
              precision    recall  f1-score   support

           0      0.848     0.718     0.778        39
           1      0.908     0.956     0.931       113

    accuracy                          0.895       152
   macro avg      0.878     0.837     0.854       152
weighted avg      0.892     0.895     0.892       152

Macierz pomyłek:
 [[ 28  11]
 [  5 108]]
```

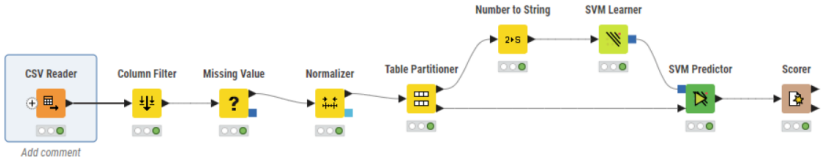


[12]:

```
# === 7. Regresja liniowa
linreg = Pipeline(steps=[("prep", num_tf_r), ("reg", LinearRegression())])

# CV informacyjnie
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
cv_r2 = cross_val_score(linreg, Xtr_r, ytr_r, cv=kfold, scoring="r2")
print("\n=== Regresja liniowa (CV, R²) ===")
print(cv_r2, "średnio:", np.mean(cv_r2))

linreg.fit(Xtr_r, ytr_r)
yhat = linreg.predict(Xte_r)

mae = mean_absolute_error(yte_r, yhat)
mse = mean_squared_error(yte_r, yhat)
rmse = np.sqrt(mse)
r2 = r2_score(yte_r, yhat)

print("\n--- Regresja (test) ---")
print("MAE :", mae)
print("RMSE:", rmse)
print("R²  :", r2)
```

```
=== Regresja liniowa (CV, R²) ===
[0.99646326 0.99820199 0.99284827 0.9951352  0.97723988] średnio: 0.9919777192042796

--- Regresja (test) ---
MAE : 83349488.44972773
RMSE: 136266823.49197578
R²  : 0.9996830552982623
```

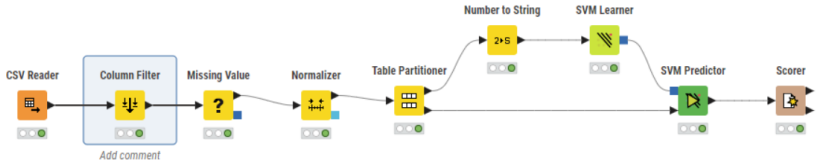




▶ 1: File Table Flow Variables

Rows: 756 | Columns: 755 Table | Statistics

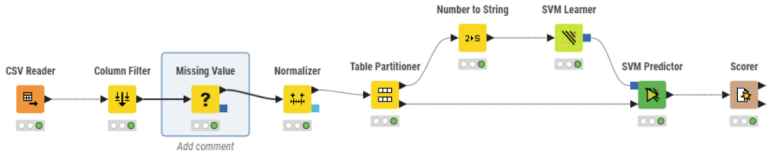
#	RowID	id Number (Inte...	gender Number (Inte...	PPE Number (Float)	DFA Number (Float)	RPDE Number (Float)	numPulses Number (Inte...	numPerio... Number (Inte...	meanPeri... Number (Float)	stdDevPe... Number (Float)	locPctJitt... Number (Float)	locAbsJit... Number (Floa
5	Row4	1	0	0.328	0.798	0.53	236	235	0.008	0.003	0.005	0
6	Row5	1	0	0.508	0.787	0.655	226	221	0.008	0.003	0.008	0
7	Row6	2	1	0.761	0.621	0.545	322	321	0.006	0	0.002	0
8	Row7	2	1	0.837	0.621	0.512	318	317	0.006	0	0.003	0
9	Row8	2	1	0.808	0.618	0.504	318	317	0.006	0	0.002	0
10	Row9	3	0	0.853	0.622	0.549	493	492	0.004	0	0.001	0
11	Row10	3	0	0.807	0.673	0.617	488	487	0.004	0	0.001	0
12	Row11	3	0	0.827	0.583	0.446	498	497	0.004	0	0.001	0
13	Row12	4	0	0.873	0.79	0.78	492	491	0.004	0	0.003	0
14	Row13	4	0	0.811	0.768	0.708	305	304	0.006	0.003	0.003	0



▶ 1: Filtered table Flow Variables

Rows: 756 | Columns: 755 Table | Statistics

#	RowID	id Number (Inte...	gender Number (Inte...	PPE Number (Float)	DFA Number (Float)	RPDE Number (Float)	numPulses Number (Inte...	numPerio... Number (Inte...	meanPeri... Number (Float)	stdDevPe... Number (Float)	locPctJitt... Number (Float)	locAbsJit Number (Flo
1	Row0	0	1	0.852	0.718	0.572	240	239	0.008	0	0.002	0
2	Row1	0	1	0.767	0.695	0.54	234	233	0.008	0	0.002	0
3	Row2	0	1	0.851	0.676	0.59	232	231	0.008	0	0.002	0
4	Row3	1	0	0.411	0.797	0.593	178	177	0.011	0	0.004	0
5	Row4	1	0	0.328	0.798	0.53	236	235	0.008	0.003	0.005	0
6	Row5	1	0	0.508	0.787	0.655	226	221	0.008	0.003	0.008	0
7	Row6	2	1	0.761	0.621	0.545	322	321	0.006	0	0.002	0
8	Row7	2	1	0.837	0.621	0.512	318	317	0.006	0	0.003	0
9	Row8	2	1	0.808	0.618	0.504	318	317	0.006	0	0.002	0
10	Row9	3	0	0.853	0.622	0.549	493	492	0.004	0	0.001	0

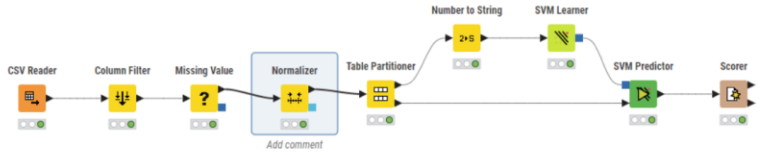


▶ 1: Output table ■ 2: PMML Transformations Flow Variables

Rows: 756 | Columns: 755

Table Statistics

	#	RowID	id Number (Float)	gender Number (Float)	PPE Number (Float)	DFA Number (Float)	RPDE Number (Float)	numPulses Number (Float)	numPerio... Number (Float)	meanPeri... Number (Float)	stdDevPe... Number (Float)	locPctJitt... Number (Float)	locAbsJ Number (F
☐	1	Row0	0	1	0.852	0.718	0.572	240	239	0.008	0	0.002	0
☐	2	Row1	0	1	0.767	0.695	0.54	234	233	0.008	0	0.002	0
☐	3	Row2	0	1	0.851	0.676	0.59	232	231	0.008	0	0.002	0
☐	4	Row3	1	0	0.411	0.797	0.593	178	177	0.011	0	0.004	0
☐	5	Row4	1	0	0.328	0.798	0.53	236	235	0.008	0.003	0.005	0
☐	6	Row5	1	0	0.508	0.787	0.655	226	221	0.008	0.003	0.008	0
☐	7	Row6	2	1	0.761	0.621	0.545	322	321	0.006	0	0.002	0
☐	8	Row7	2	1	0.837	0.621	0.512	318	317	0.006	0	0.003	0
☐	9	Row8	2	1	0.808	0.618	0.504	318	317	0.006	0	0.002	0
☐	10	Row9	3	0	0.853	0.622	0.549	493	492	0.004	0	0.001	0

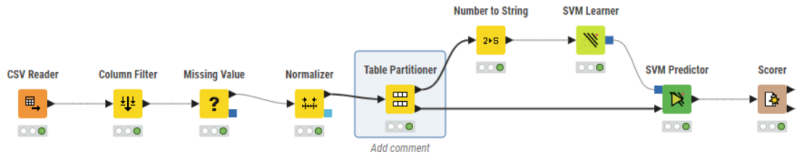


▶ 1: Normalized table ■ 2: Normalize Model Flow Variables

Rows: 756 | Columns: 755

Table Statistics

	#	RowID	id Number (Float)	gender Number (Float)	PPE Number (Float)	DFA Number (Float)	RPDE Number (Float)	numPulses Number (Float)	numPerio... Number (Float)	meanPeri... Number (Float)	stdDevPe... Number (Float)	locPctJitt... Number (Float)	locAbsJit... Number (Float)
☐	1	Row0	-1.724	0.968	0.627	0.256	0.605	-0.846	-0.842	0.933	-0.407	-0.055	0.038
☐	2	Row1	-1.724	0.968	0.122	-0.08	0.368	-0.907	-0.902	1.039	-0.426	-0.142	-0.028
☐	3	Row2	-1.724	0.968	0.618	-0.35	0.733	-0.927	-0.922	1.084	-0.443	-0.215	-0.089
☐	4	Row3	-1.71	-1.032	-1.979	1.381	0.753	-1.471	-1.466	2.463	-0.275	0.71	1.256
☐	5	Row4	-1.71	-1.032	-2.471	1.397	0.3	-0.887	-0.882	0.986	3.142	1.151	1.177
☐	6	Row5	-1.71	-1.032	-1.409	1.248	1.204	-0.987	-1.023	0.696	3.179	2.095	1.876
☐	7	Row6	-1.697	0.968	0.087	-1.133	0.41	-0.02	-0.017	-0.202	-0.379	-0.04	-0.15
☐	8	Row7	-1.697	0.968	0.534	-1.142	0.165	-0.06	-0.057	-0.157	-0.34	0.189	0.016
☐	9	Row8	-1.697	0.968	0.366	-1.187	0.112	-0.06	-0.057	-0.166	-0.431	-0.272	-0.306
☐	10	Row9	-1.683	-1.032	0.63	-1.118	0.433	1.704	1.703	-1.341	-0.471	-0.599	-0.603

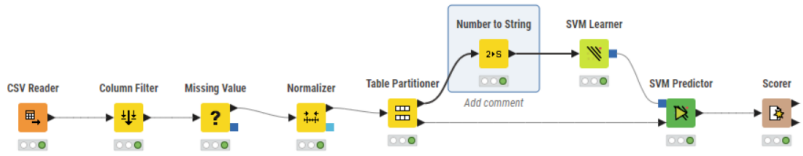


▶ 1: First partition ▶ 2: Second partition Flow Variables

Rows: 529 | Columns: 755

Table Statistics

#	RowID	id Number (Float)	gender Number (Float)	PPE Number (Float)	DFA Number (Float)	RPDE Number (Float)	numPulses Number (Float)	numPerio... Number (Float)	meanPeri... Number (Float)	stdDevPe... Number (Float)	locPctJitt... Number (Float)	locAbsJit... Number (Float)
1	Row0	-1.724	0.968	0.627	0.256	0.605	-0.846	-0.842	0.933	-0.407	-0.055	0.038
2	Row1	-1.724	0.968	0.122	-0.08	0.368	-0.907	-0.902	1.039	-0.426	-0.142	-0.028
3	Row2	-1.724	0.968	0.618	-0.35	0.733	-0.927	-0.922	1.084	-0.443	-0.215	-0.089
4	Row3	-1.71	-1.032	-1.979	1.381	0.753	-1.471	-1.466	2.463	-0.275	0.71	1.256
5	Row4	-1.71	-1.032	-2.471	1.397	0.3	-0.887	-0.882	0.986	3.142	1.151	1.177
6	Row6	-1.697	0.968	0.087	-1.133	0.41	-0.02	-0.017	-0.202	-0.379	-0.04	-0.15
7	Row7	-1.697	0.968	0.534	-1.142	0.165	-0.06	-0.057	-0.157	-0.34	0.189	0.016
8	Row9	-1.683	-1.032	0.63	-1.118	0.433	1.704	1.703	-1.341	-0.471	-0.599	-0.603
9	Row10	-1.683	-1.032	0.356	-0.4	0.934	1.653	1.653	-1.316	-0.452	-0.569	-0.587
10	Row11	-1.683	-1.032	0.474	-1.68	-0.317	1.754	1.754	-1.362	-0.481	-0.622	-0.614

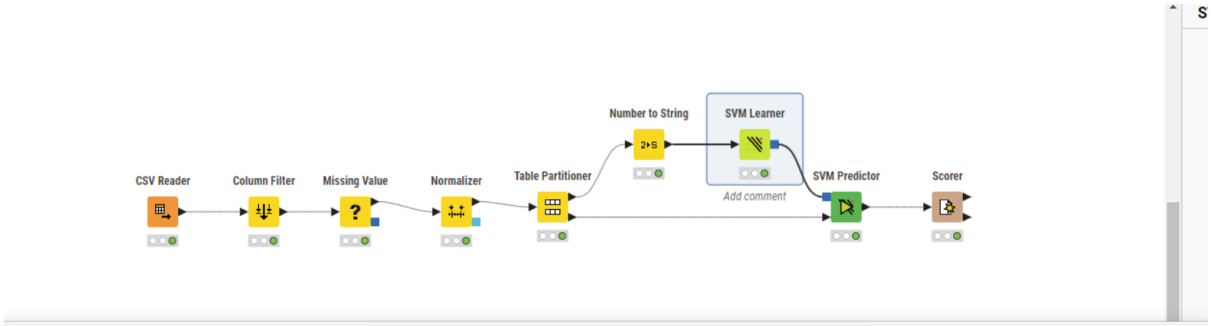


▶ 1: Transformed input Flow Variables

Rows: 529 | Columns: 755

Table Statistics

#	RowID	id Number (Float)	gender Number (Float)	PPE Number (Float)	DFA Number (Float)	RPDE Number (Float)	numPulses Number (Float)	numPerio... Number (Float)	meanPeri... Number (Float)	stdDevPe... Number (Float)	locPctJitt... Number (Float)	locAbsJit... Number (Float)
1	Row0	-1.724	0.968	0.627	0.256	0.605	-0.846	-0.842	0.933	-0.407	-0.055	0.038
2	Row1	-1.724	0.968	0.122	-0.08	0.368	-0.907	-0.902	1.039	-0.426	-0.142	-0.028
3	Row2	-1.724	0.968	0.618	-0.35	0.733	-0.927	-0.922	1.084	-0.443	-0.215	-0.089
4	Row3	-1.71	-1.032	-1.979	1.381	0.753	-1.471	-1.466	2.463	-0.275	0.71	1.256
5	Row4	-1.71	-1.032	-2.471	1.397	0.3	-0.887	-0.882	0.986	3.142	1.151	1.177
6	Row6	-1.697	0.968	0.087	-1.133	0.41	-0.02	-0.017	-0.202	-0.379	-0.04	-0.15
7	Row7	-1.697	0.968	0.534	-1.142	0.165	-0.06	-0.057	-0.157	-0.34	0.189	0.016
8	Row9	-1.683	-1.032	0.63	-1.118	0.433	1.704	1.703	-1.341	-0.471	-0.599	-0.603
9	Row10	-1.683	-1.032	0.356	-0.4	0.934	1.653	1.653	-1.316	-0.452	-0.569	-0.587
10	Row11	-1.683	-1.032	0.474	-1.68	-0.317	1.754	1.754	-1.362	-0.481	-0.622	-0.614



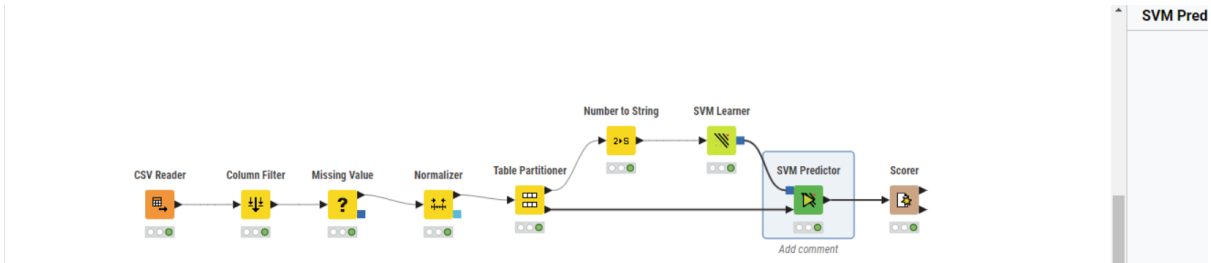
Trained SVM - 4:6 - SVM Learner

File

PMML: SupportVectorMachineModel | Spec - Columns: 754 | Flow Variables

Expand All | Collapse All

```
PMML version="4.2" xmlns="http://www.dmg.org/PMML-4_2"
  Header copyright="Administrator"
  DataDictionary numberOfFields="755"
  SupportVectorMachineModel modelName="SVM" functionName="classification" algorithmName="Sequential Minimal Optimization (SMO)" svmRepresentation="SupportVectors"
```

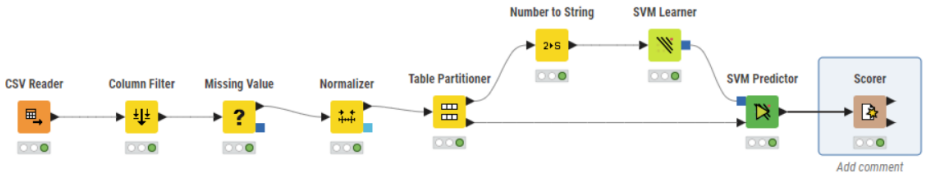


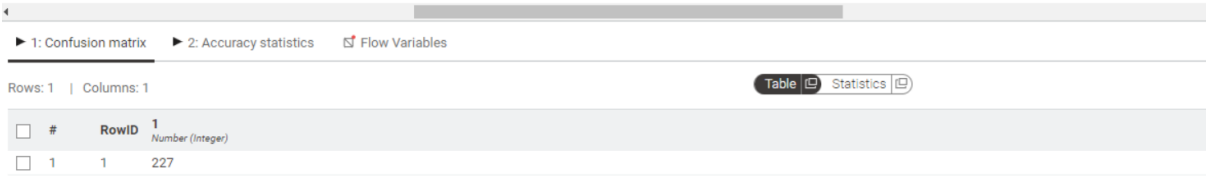
1: Classification | Flow Variables

Rows: 227 | Columns: 756

Table | Statistics

#	RowID	id Number (Float)	gender Number (Float)	PPE Number (Float)	DFA Number (Float)	RPDE Number (Float)	numPulses Number (Float)	numPerio... Number (Float)	meanPeri... Number (Float)	stdDevPe... Number (Float)	locPctJitt... Number (Float)	locAbsJit... Number (Float)	rapJitter Number (Float)
1	Row5	-1.71	-1.032	-1.409	1.248	1.204	-0.987	-1.023	0.696	3.179	2.095	1.876	1.749
2	Row8	-1.697	0.968	0.366	-1.187	0.112	-0.06	-0.057	-0.166	-0.431	-0.272	-0.306	-0.342
3	Row12	-1.669	-1.032	0.746	1.284	2.119	1.694	1.693	-1.334	-0.434	0.181	-0.25	0.168
4	Row13	-1.669	-1.032	0.385	0.974	1.594	-0.191	-0.188	-0.024	3.934	0.413	0.212	0.331
5	Row14	-1.669	-1.032	0.375	1.14	1.475	-0.332	-0.329	0.145	3.262	0.854	0.592	1.004
6	Row19	-1.642	0.968	0.304	0.166	-0.887	-0.352	-0.349	0.182	-0.437	-0.42	-0.373	-0.413
7	Row22	-1.628	0.968	0.094	0.497	1.947	-1.501	-1.586	1.816	3.897	3.94	4.628	4.42
8	Row23	-1.628	0.968	-2.105	1.13	1.279	-0.554	-0.55	0.451	-0.395	0.299	0.243	0.25
9	Row26	-1.614	0.968	0.451	-1.013	0.534	-0.352	-0.349	0.155	-0.419	-0.348	-0.323	-0.362
10	Row28	-1.6	0.968	0.53	-1.255	-0.392	0.091	0.094	-0.304	-0.445	-0.375	-0.39	-0.423





3. Wnioski

Przeprowadzona analiza wykazała, że zastosowanie klasyfikatora **SVM** pozwala skutecznie rozróżniać przypadki choroby Parkinsona na podstawie cech akustycznych mowy. Model uzyskał wysoką wartość metryk takich jak **Accuracy** i **F1-score**, co wskazuje na jego dobrą ogólną skuteczność w detekcji choroby. Zastosowanie walidacji krzyżowej potwierdziło stabilność wyników i brak nadmiernego dopasowania.

W przypadku modelu **regresji liniowej**, uzyskane wartości błędów (MAE, RMSE) oraz współczynnik determinacji $\mathbf{R^2}$ wskazują, że relacja między wybraną zmienną ciągłą a pozostałymi cechami ma charakter umiarkowanie liniowy. Wyniki są poprawne, jednak model liniowy nie zawsze jest w stanie uchwycić nieliniowe zależności w danych akustycznych.

Porównanie obu modeli potwierdziło, że klasyfikacja choroby Parkinsona jest lepiej dopasowanym zadaniem dla dostępnych danych niż regresja, a metoda **SVM z odpowiednim doborem jądra (np. RBF)** daje najlepsze rezultaty. Dalsze ulepszenia mogłyby obejmować selekcję najistotniejszych cech, zastosowanie metod nieliniowych dla regresji (np. SVR lub Random Forest Regressor) oraz techniki balansowania klas w przypadku nierównowagi danych.