تشخیص الگوهای حرکتی با داده‌های سنسوری موبایل

پیاده‌سازی خط لوله یادگیری ماشین: از کاهش ابعاد با PCA تا تنظیم ابرپارامترها با Optuna

ارشیا کلانتریان
۴۰۱۲۱۹۹۳

محمد حسین گل محمدی
۴۰۲۲۰۹۰۳

معرفی موضوع

- این پروژه بر روی **تشخیص هوشمند فعالیت‌های انسان** (HAR) تمرکز دارد که شامل شناسایی حرکاتی مانند راه رفتن، نشستن، ایستادن و دراز کشیدن است.
- داده‌های مورد استفاده از سنسورهای **شتاب‌سنج** و **ژیروسکوپ** موجود در گوشی‌های هوشمند استخراج شده‌اند.

هدف اصلی

- طراحی یک سیستم یادگیری ماشین که بتواند با دقت بالا، نوع فعالیت فرد را تشخیص دهد.
- کاهش پیچیدگی محاسباتی با استفاده از روش PCA جهت آماده‌سازی مدل برای اجرا در سیستم‌های با منابع محدود.
- مقایسه هوشمند دو الگوریتم SVM و MLP برای پیدا کردن بهینه‌ترین مدل از نظر سرعت و دقت.

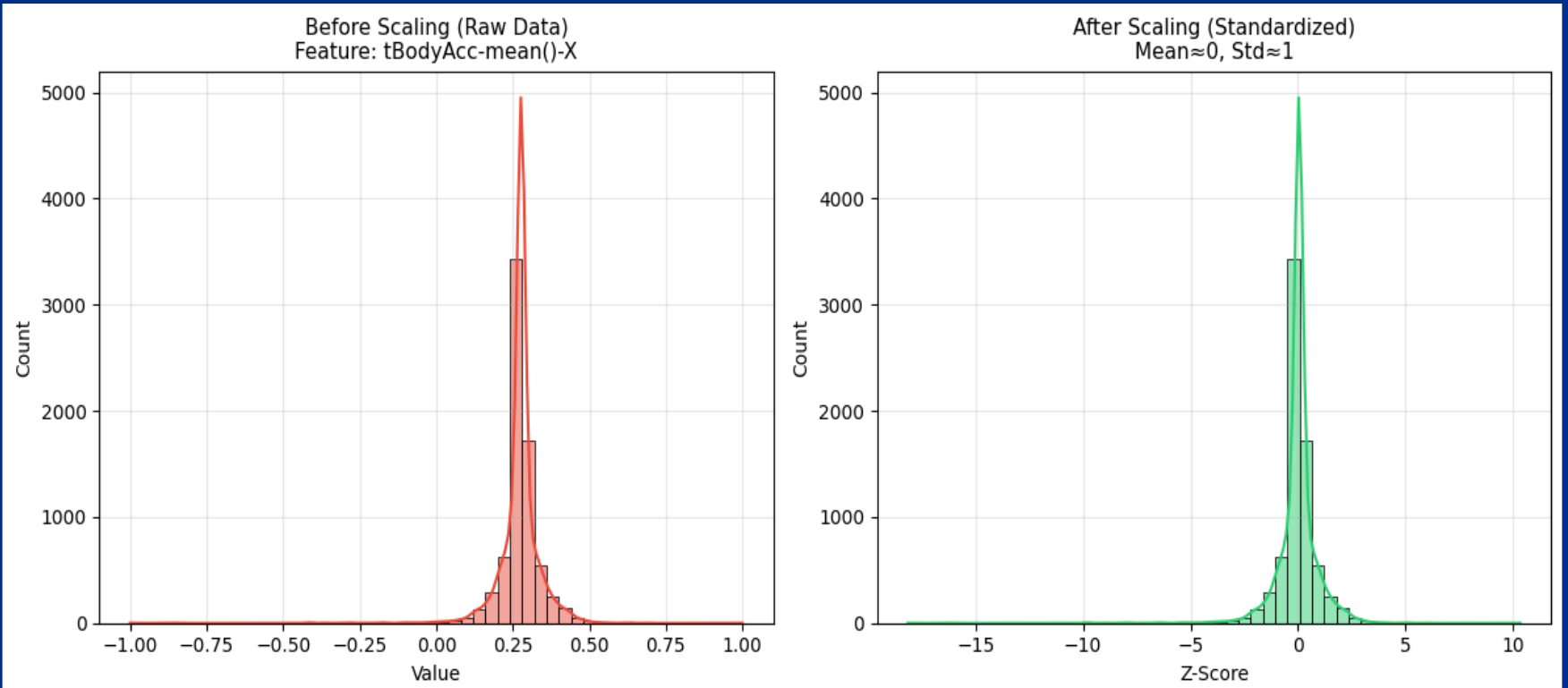
آشنایی با داده‌های خام و آماده‌سازی اولیه

Dataset

- نام دیتابیس: UCI Human Activity Recognition
- منبع :داده‌های سنسورهای شتاب‌سنج و ژیروسکوپ گوشی هوشمند.
- تعداد ویژگی‌ها: 561 ویژگی استخراج شده در حوزه‌های زمان و فرکانس.
- تعداد کلاس‌ها: 6 فعالیت (نشستن، ایستادن، دراز کشیدن، راه رفتن، بالا رفتن از پله و پایین آمدن).

Standardization

- چالش: داده‌های سنسورهای مختلف بازه‌های عددی متفاوتی داشتند.
- راهکار: استفاده از StandardScaler
- نتیجه: تمام ویژگی‌ها دارای میانگین و انحراف معیار 1 شدند تا مدل بتواند بدون Bias یاد بگیرد.



استاندارد سازی داده ها

کاهش ابعاد با استفاده از PCA

ضرورت کاهش ابعاد

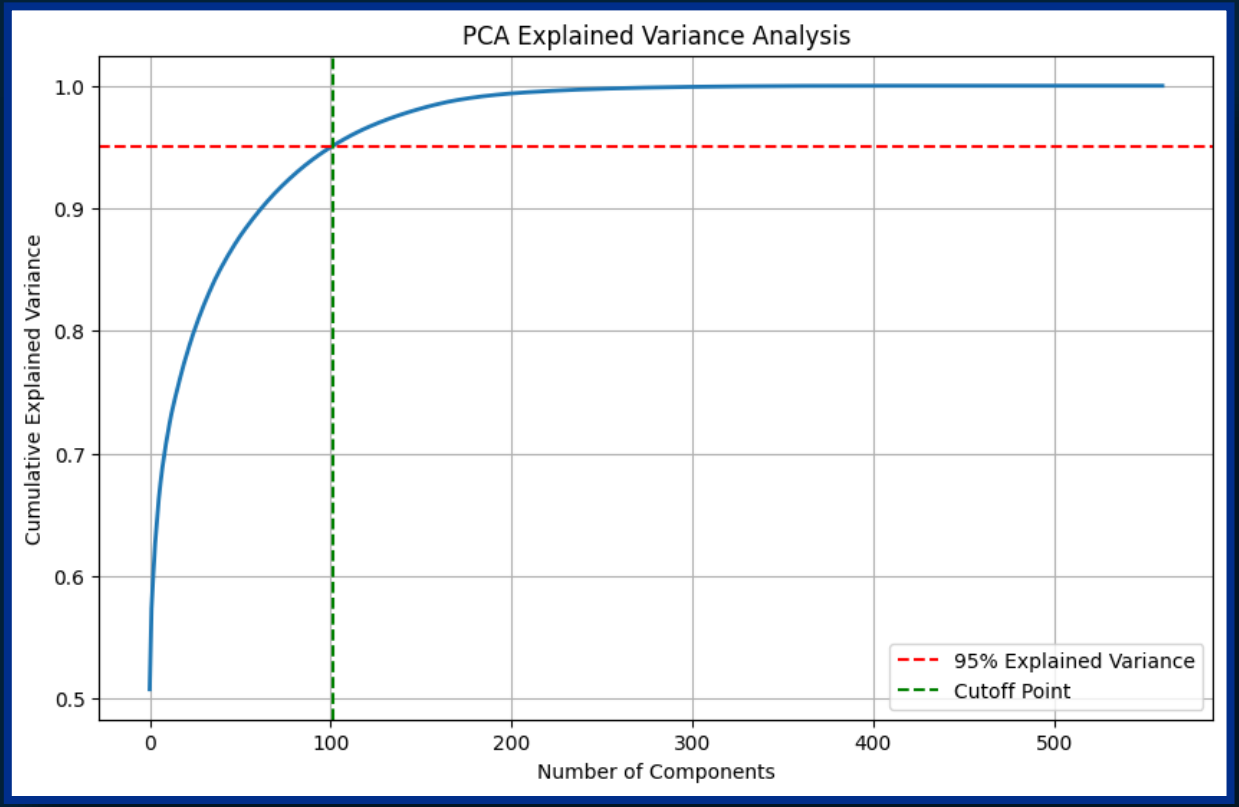
- داده‌های اولیه شامل 561 ویژگی بودند که پردازش آن‌ها باعث کندی مدل و اشغال حافظه زیاد می‌شد.
- بسیاری از سنسورها اطلاعات مشابهی را ثبت می‌کردند که باعث ایجاد نویز در یادگیری می‌شد.

پروسه حفظ اطلاعات

- هدف :استخراج ویژگی‌های جدیدی که بیشترین واریانس را در خود جای داده‌اند.
- معیار انتخاب : حفظ 95% از واریانس کل داده‌ها.

نتایج

- کاهش تعداد ویژگی‌ها از 561 به 102 (حدود 82% کاهش حجم داده‌ها).
- افزایش حدود 4 برابری سرعت آموزش و استنتاج مدل.



کاهش ابعاد به کمک PCA

تحلیل معماری مدل‌های انتخابی

مدل SVM :

- استفاده از اصل Structural Risk Minimization برای یافتن ابرصفحه‌ای (Hyper Plane) که بیشترین حاشیه (Margin) را بین کلاس‌ها ایجاد کند.
- پایداری بالا در فضاهای با ابعاد بالا و مقاومت در برابر Overfitting در داده‌های جدولی.
- SVM به دنبال نظم ساختاری است

تحلیل معماری مدل‌های انتخابی

مدل MLP :

- استفاده از Universal Approximation Theorem این شبکه می‌تواند هر تابع پیچیده و غیرخطی را که بین حرکات بدن و خروجی سنسور وجود دارد، مدل‌سازی کند.

- استفاده از توابع فعال‌ساز غیرخطی (مانند ReLU یا Tanh) برای استخراج ویژگی‌های مرتبه بالا از سیگنال‌های زمانی.

- MLP به دنبال یادگیری الگوهای پنهان و غیرخطی در داده‌هاست.

Optimization به کمک Optuna

روش جستجو (TPE Algorithm)

استفاده از الگوریتم Tree-structured Parzen Estimator برخلاف Grid Search که تمام نقاط را کورکورانه تست می‌کند این روش یک مدل احتمالی از فضای پارامترها می‌سازد.

Exploration vs. Exploitation

سیستم ابتدا فضا را کاوش می‌کند و سپس روی نواحی که (Accuracy) بیشتری گزارش کرده‌اند، متمرکز می‌شود.

این کار باعث می‌شود مدل ما نسبت به داده‌های خاص حساس نباشد و قدرت Generalization بالایی پیدا کند.

```python
# ------------------------------------------------------------
# الف) بهینه‌سازی مدل SVM (مدل کلاسیک)
# ------------------------------------------------------------
def objective_svm(trial):
    # تعریف فضای جستجو برای پارامترها
    # C: پارامتر جریمه (بین 0.1 تا 100 به صورت لگاریتمی)
    c = trial.suggest_float("C", 0.1, 100, log=True)
    # Kernel: نوع هسته (خطی یا RBF)
    kernel = trial.suggest_categorical("kernel", ["linear", "rbf"])
    # Gamma: ضریب هسته (فقط برای RBF است مهم)
    gamma = trial.suggest_categorical("gamma", ["scale", "auto"])

    # ساخت مدل با پارامترهای پیشنهادی Optuna
    model = SVC(C=c, kernel=kernel, gamma=gamma, random_state=SEED)

    # ارزیابی مدل با Cross-Validation (۵ لایه)
    # استفاده می‌کنیم چون سرعتش ۴ برابر بود (PCA) از داده‌های کاهش‌یافته
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=SEED)
    scores = cross_val_score(model, X_train_pca, y_train_enc, cv=cv, scoring="accuracy", n_jobs=-1)

    return scores.mean()

print("\n-> Tuning SVM...")
study_svm = optuna.create_study(direction="maximize")
study_svm.optimize(objective_svm, n_trials=N_TRIALS)

print(f"Best SVM Params: {study_svm.best_params}")
print(f"Best SVM Accuracy: {study_svm.best_value:.4f}")
```

کد بهینه سازی Optuna برای SVM

```python
# ------------------------------------------------------------
# ب) بهینه‌سازی مدل MLP (Deep Learning)
# ------------------------------------------------------------
def objective_mlp(trial):
    # تعریف تعداد نورون‌ها در لایه‌های مخفی
    # مثلاً (50,) یعنی یک لایه ۵۰ تایی. (100, 50) یعنی دو لایه.
    n_layers = trial.suggest_int("n_layers", 1, 3) # تعداد لایه‌ها بین ۱ تا ۳
    layers = []
    for i in range(n_layers):
        layers.append(trial.suggest_int(f"n_units_l{i}", 30, 150)) # تعداد نورون هر لایه

    # نرخ یادگیری (Learning Rate)
    lr_init = trial.suggest_float("learning_rate_init", 1e-4, 1e-2, log=True)
    # آلفا (Regularization) برای جلوگیری از Overfitting
    alpha = trial.suggest_float("alpha", 1e-5, 1e-2, log=True)
    # تابع فعال‌ساز
    activation = trial.suggest_categorical("activation", ["relu", "tanh"])

    model = MLPClassifier(
        hidden_layer_sizes=tuple(layers),
        learning_rate_init=lr_init,
        alpha=alpha,
        activation=activation,
        max_iter=500, # تعداد دورهای آموزش
        random_state=SEED,
        early_stopping=True # اگر پیشرفت نکرد، متوقف شو (برای سرعت)
    )

    cv = StratifiedKFold(n_splits=3, shuffle=True, random_state=SEED)
    scores = cross_val_score(model, X_train_pca, y_train_enc, cv=cv, scoring="accuracy", n_jobs=-1)

    return scores.mean()

print("\n-> Tuning MLP (Deep Learning)...")
study_mlp = optuna.create_study(direction="maximize")
study_mlp.optimize(objective_mlp, n_trials=N_TRIALS)

print(f"Best MLP Params: {study_mlp.best_params}")
print(f"Best MLP Accuracy: {study_mlp.best_value:.4f}")
```

کد بهینه سازی Optuna برای MLP

نتایج بهینه سازی با Optuna

SVM

- C: 2.45
- Kernel : Linear
- Gamma: Scale
- Accuracy: 97.37%

MLP

- Number of layers: 1
- Units: 143
- Activation: ReLU
- Accuracy: 97.57%

تحلیل نتایج بهینه سازی با Optuna

SVM

- C: 2.45
- Kernel : Linear
- Gamma: Scale
- Test Accuracy: 92.3%

پارامتر C

عدد ۲,۴۵ نشان می‌دهد که ما یک Soft Margin در نظر گرفته شده.
یعنی مدل نه آنقدر سخت‌گیر است که داده‌های نویز را حفظ کند
Overfitting و نه آنقدر ساده‌گیر که الگوها را نبیند.

پارامتر Linear

انتخاب کرنل خطی ثابت کرد که مرحله کاهش ابعاد آنقدر داده‌ها را تمیز
و مرتب کرده که برای جدا کردن کلاس‌ها نیازی به استفاده از RBF
نداریم. این یعنی سرعت بسیار بالا

پارامتر Scale

تنظیم خودکار بر اساس واریانس داده‌ها
هرچند در کرنل خطی گاما نقش کلیدی ندارد، اما تنظیم آن روی
Scale باعث شد داده‌ها قبل از پردازش نرمالایز شوند

تحلیل نتایج بهینه سازی با Optuna

MLP

- Number of layers: 1
- Units: 143
- Activation: ReLU
- Test Accuracy: 93.48%

Single layer

Optuna تشخیص داد که یک لایه مخفی برای استخراج الگوهای این سنسورها کاملاً کافی بود و افزایش لایه ها زمان محاسبات را بالا می برد

143 Neurons

در کل ۱۰۲ ویژگی ورودی داشتیم. شبکه این تعداد را به ۱۴۳ نورون رساند تا با افزایش ابعاد، جزئیات ریزتر را بهتر تشخیص دهد

ReLU

استفاده از ReLU باعث شد مقادیر منفی یعنی همان سیگنال‌های غیرمهم صفر شوند. این کار شبکه را Sparse و سبک کرد و باعث شد دقت نهایی از SVM هم بالاتر برود.

چرا روش Grid Search استفاده نشد؟

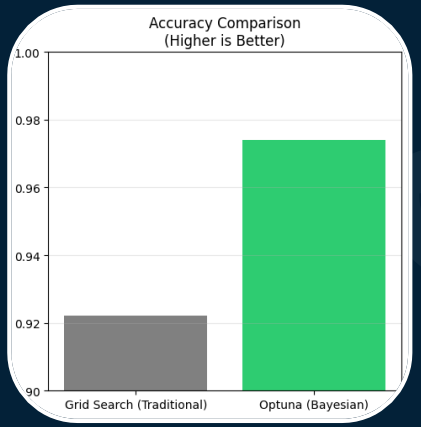
زمان بر بودن و محاسبات بیشتر

Grid Search:
524s ~ (9 mins)

Optuna:
209s ~ (3.5 mins)

روش Optuna، 60% سریع‌تر به جواب رسید.

Grid Search تمام خانه‌ها را کورکورانه چک می‌کند، اما Optuna مسیرهای بد را زودتر رها می‌کند



رزولوشن و نقاط کور

Grid Search:
~ 92.22%

Optuna:
~ 97.37%

Grid Search فقط نقاط خاصی را می‌بیند مثلا فقط C = 1 یا C = 10 و چون بهترین C = 2.45 بود دیده نشد اما Optuna چون در فضای پیوسته جستجو میکند دقت بهتری داشت

معماری آموزش و تضمین صحت مدل

تقسیم داده ها

استفاده از ۷۳۵۲ نمونه برای یادگیری یعنی تقریبا 70%
و ۲۹۴۷ نمونه برای تست یعنی تقریبا 30%

صحت سنجی به روش Fold Cross-Validation-5

تقسیم داده های آموزش به ۵ و آموزش مدل روی ۴ بخش و اعتبار سنجی روی بخش
پنجم که باعث پیدا کردن بهترین میانگین عملکرد در کل داده ها شود

ایجاد تکرار پذیری

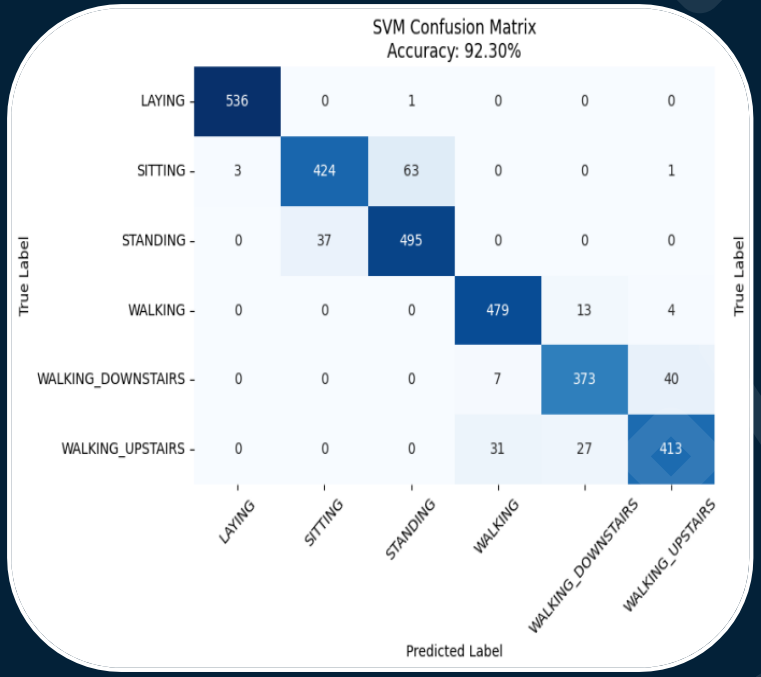
با ثابت نگه داشتن مقدار Seed روی 42 تضمین میکند که تمام تقسیم بندی ها و
وزن های اولیه در ران های بعدی ثابت می ماند
با اعمال StandardScalar بر هر دو بخش Test و Train به صورت مجزا از
data leakage جلوگیری شد

```python
# ---------------------------------------------------------
# ۱۰. ساخت مدل‌های نهایی با پارامترهای بهینه (از مرحله قبل)
# ---------------------------------------------------------
print("6. Final Training & Evaluation...")

# مقادیر جدید برای کد نهایی
final_svm = SVC(C=2.45, kernel='linear', probability=True, random_state=SEED)

final_mlp = MLPClassifier(
    hidden_layer_sizes=(143,),  # تعداد نورون جدید
    activation='relu',          # اکتیویشن جدید
    learning_rate_init=0.0053,  # لرنینگ ریت جدید
    alpha=0.00021,              # آلفای جدید
    max_iter=1000,
    random_state=SEED
)

# ---------------------------------------------------------
# ۲۰. آموزش روی داده‌های Train و پیش‌بینی روی Test
# ---------------------------------------------------------
print("-> Training Final Models...")
final_svm.fit(X_train_pca, y_train_enc)
y_pred_svm = final_svm.predict(X_test_pca)

final_mlp.fit(X_train_pca, y_train_enc)
y_pred_mlp = final_mlp.predict(X_test_pca)
```

```python
# ---------------------------------------------------------
# ۲۰. گزارش متنی نتایج (Classification Report)
# ---------------------------------------------------------
print("\n" + "="*40)
print("FINAL RESULTS: SVM (Linear)")
print("="*40)
print(classification_report(y_test_enc, y_pred_svm, target_names=le.classes_))
acc_svm = accuracy_score(y_test_enc, y_pred_svm)
print(f"SVM Test Accuracy: {acc_svm:.2%}")

print("\n" + "="*40)
print("FINAL RESULTS: MLP (Deep Learning)")
print("="*40)
print(classification_report(y_test_enc, y_pred_mlp, target_names=le.classes_))
acc_mlp = accuracy_score(y_test_enc, y_pred_mlp)
print(f"MLP Test Accuracy: {acc_mlp:.2%}")

# ---------------------------------------------------------
# ۴۰. تصویرسازی: ماتریس درهم‌ریختگی (Section 7)
# ---------------------------------------------------------
print("\n7. Visualization (Confusion Matrix)...")

def plot_confusion_matrix(y_true, y_pred, title, ax):
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=le.classes_, yticklabels=le.classes_, ax=ax)
    ax.set_title(title, fontsize=12)
    ax.set_ylabel('True Label')
    ax.set_xlabel('Predicted Label')
    ax.tick_params(axis='x', rotation=45)

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

plot_confusion_matrix(y_test_enc, y_pred_svm, f'SVM Confusion Matrix\nAccuracy: {acc_svm:.2%}', axes[0])
plot_confusion_matrix(y_test_enc, y_pred_mlp, f'MLP Confusion Matrix\nAccuracy: {acc_mlp:.2%}', axes[1])

plt.tight_layout()
plt.show()
```

کد آموزش و صحت سنجی داده ها

تحلیل عملکرد مدل SVM



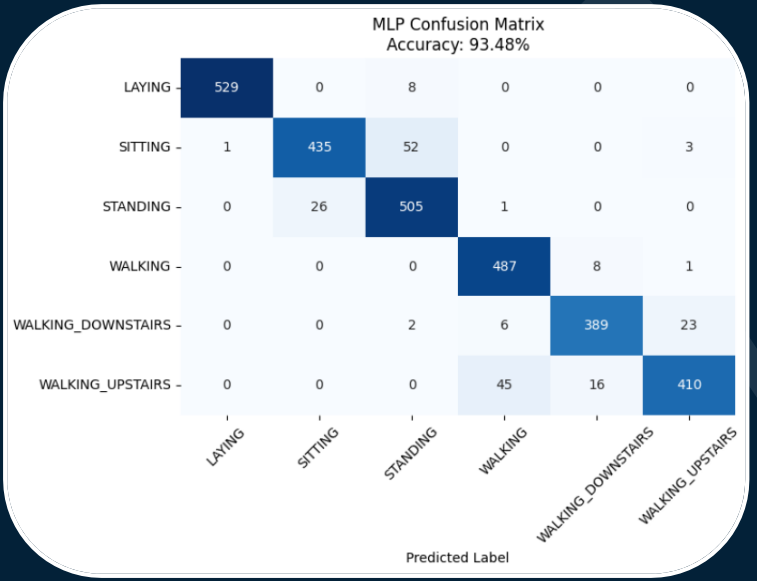
- مدل SVM در تفکیک Sitting و Standing با دقت بسیار بالایی عمل کرده اما حدود 63 مورد خطا بین این دو کلاس دارد.

  با وجود استفاده از کرنل خطی به دلیل استاندارد سازی دمل توانسته مرز های بهینه ای پیدا کند و به دقت 92.3% برسد

- تشخیص ۱۰۰ درصدی فعالیت‌های Walking و Laying

  مدل به دلیل سادگی سرعت بسیار بالایی دارد

تحلیل عملکرد مدل MLP



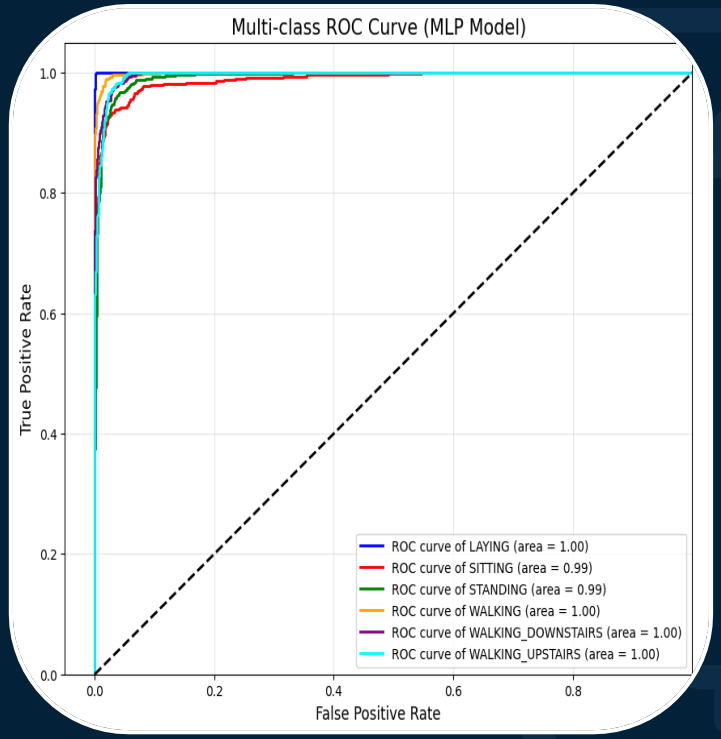
- مدل MLP با دقت 93.48% حدود 1.2% از SVM دقیق‌تر عمل کرده است. استفاده از ۱۴۳ نورون و تابع فعال‌ساز ReLU به شبکه اجازه داده تا الگوهای غیرخطی بسیار ظریف در لرزش‌های بدن را بهتر از SVM شناسایی کند.
- اگر به ماتریس نگاه کنیم، MLP تعداد خطاهای کمتری در تفکیک Sitting از Standing نسبت به SVM دارد
- با وجود دقت بالاتر این مدل به دلیل لایه‌های عصبی، حافظه و پردازش بیشتری نسبت به SVM مصرف می‌کند.

مقایسه نتایج دو مدل

ویژگی	SVM	MLP
Accuracy	92.3%	93.48
پیچیدگی	کمتر	بیشتر
تشخیص ایستا	خوب	عالی به دلیل نگاشت ReLU

منحنی ROC و تحلیل قدرت تفکیک

مفهوم نمودار ROC و AUC



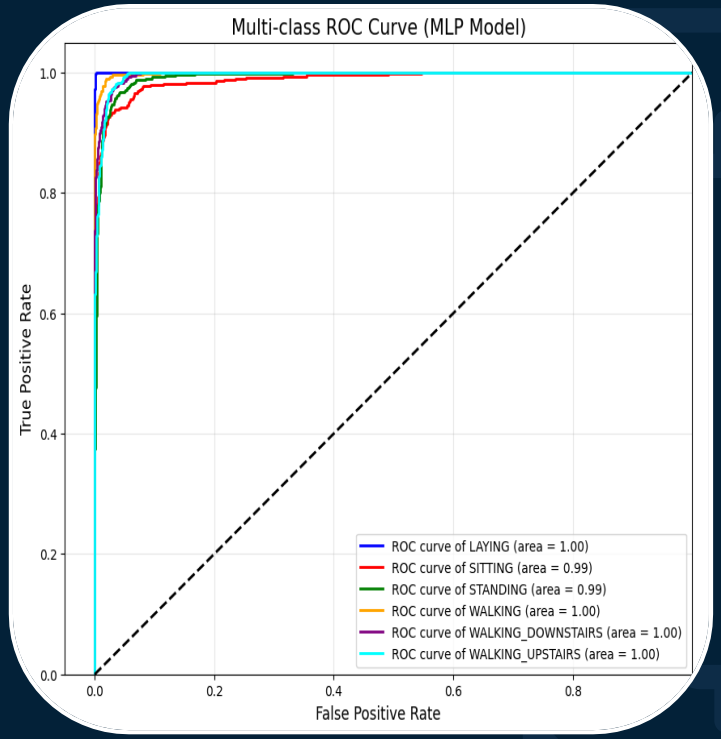
نمودار ROC نرخ مثبت واقعی Sensitivity را در مقابل نرخ مثبت کاذب نشان می‌دهد.

معیار AUC مساحت زیر منحنی که عددی بین 0.5 به معنی تصادفی تا 1 به معنی عالی است.

مشاهده می‌کنیم که تمام منحنی‌ها به گوشه بالا و سمت چپ چسبیده‌اند که نشان‌دهنده عملکرد فوق‌العاده مدل است

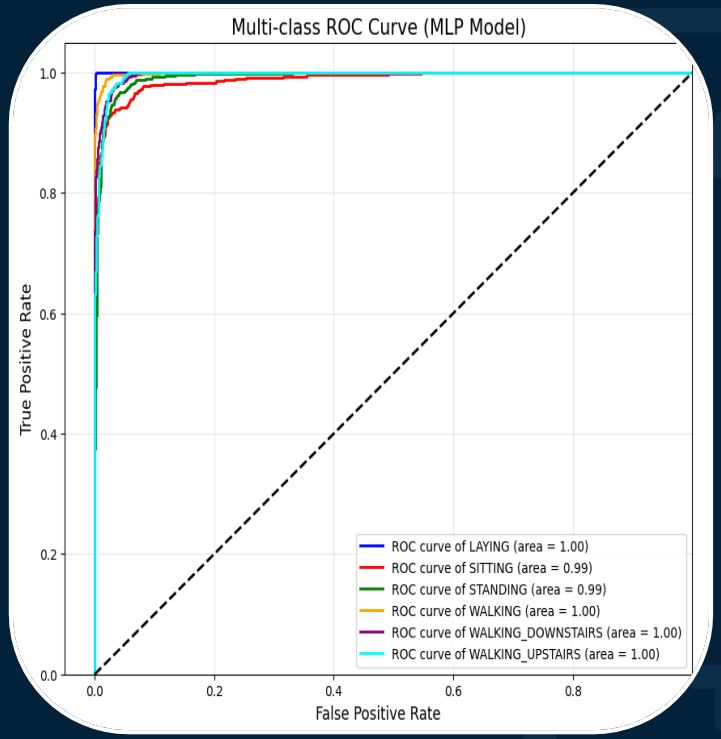
منحنی ROC و تحلیل قدرت تفکیک

تحلیل چندکلاسه



- کلاس ها تفکیک شده و برای هر ۶ فعالیت، یک منحنی مجزا ترسیم شده است
- مقادیر AUC برای اکثر کلاس‌ها (مخصوصاً Laying و Walking) برابر با 1 یا بسیار نزدیک به آن است
- پایداریِ کلی مدل در تمامی کلاس‌ها یکسان است و مدل دچار تبعیض بین فعالیت‌ها نشده است.

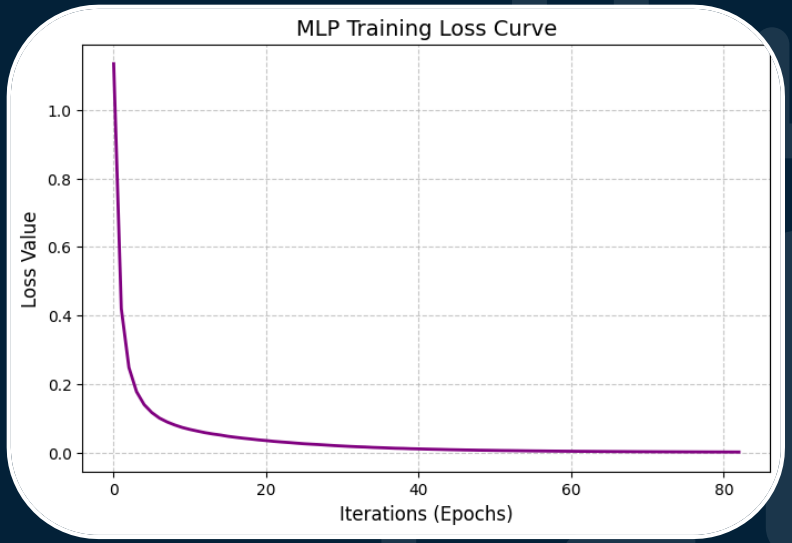
منحنی ROC و تحلیل قدرت تفکیک



تفسیر نهایی

- حتی برای کلاس‌های چالش‌برانگیز (Sitting/Standing) ، مساحت زیر منحنی بسیار بالاست.
- این یعنی مدل در رتبه‌بندی احتمالات بسیار دقیق است و حتی اگر در لبه‌ی تصمیم‌گیری اشتباه کند، فعالیت درست را با احتمال بسیار بالایی در اولویت‌های بعدی خود دارد.

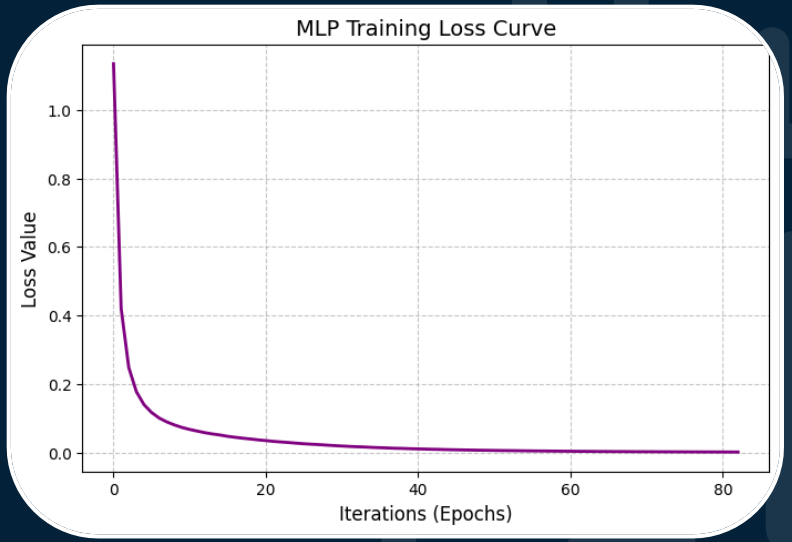
تحلیل منحنی خطا



مفهوم

- نشان‌دهنده میزان اختلاف بین پیش‌بینی مدل و فعالیت واقعی کاربر در هر مرحله از آموزش است.
- هدف :کمینه کردن این مقدار تا حد ممکن از طریق بهینه‌سازی وزن‌های شبکه عصبی.

تحلیل منحنی خطا

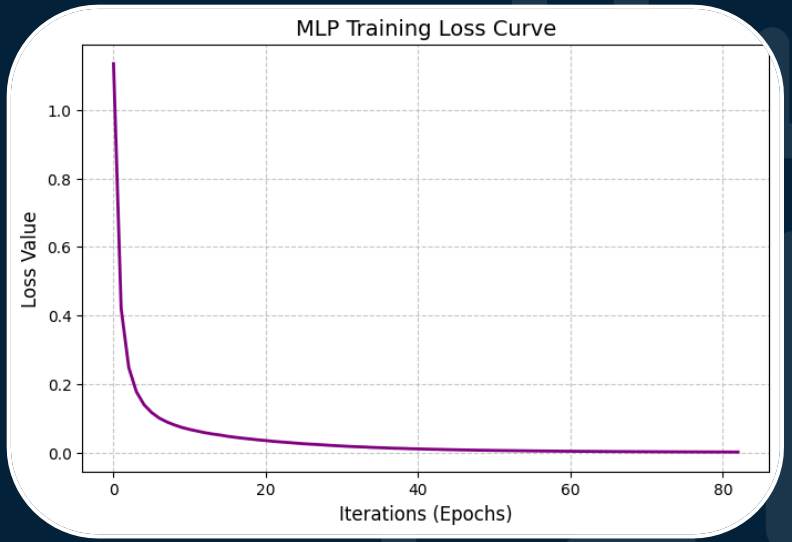


تحلیل رفتار

- منحنی به صورت آرام و بدون نوسانات شدید به پایین آمده این نشانگر انتخاب درست نرخ یادگیری است که 0.0053 بود
- اگر نرخ یادگیری بزرگتر بود منحنی دچار پرش می شد و اگر خیلی کوچک بود نمودار به این سرعت همگرا نمی شد

تحلیل منحنی خطا

پایداری و عدم Overfitting



- به ثبات رسیدن نمودار حاکی از رسیدن به نقطه بهینه است
- عدم صعود مجدد نمودار در انتهای کار یعنی اینکه مدل دچار Overfitting نشده و 143 نورون متناسب با ابعاد ورودی یعنی 102 ویژگی انتخاب شده است

نتیجه نهایی پروژه

برتری شبکه عصبی:

مدل MLP با دقت تست 93.48% توانست بهتر از مدل SVM با دقت تست 92.3% عمل کند و در تفکیک فعالیت‌های ایستا (نشستن/ایستادن) عملکرد بهتری نشان داد.

بهینه‌سازی موفق:

کاهش 82% ابعاد داده‌ها به کمک PCA و افزایش سرعت آموزش با Optuna، مدلی سبک و سریع ایجاد کرد که برای پردازش Real Time ایده‌آل است.

با تشکر از توجه شما!