وزارت علوم، تحقیقات و فناوری
موسسه آموزش عالی زند شیراز

تمرین

اینترنت اشیا

استاد

محمد زارع

به کوشش

مهدی جوانشیر

تاریخ

آبان ۱۴۰۳

عنوان تمرین نرم افزاری: پیش بینی مصرف انرژی با استفاده از یادگیری ماشین

هدف

این تمرین به شما کمک می‌کند تا با استفاده از الگوریتم‌های یادگیری ماشین، مدل‌هایی برای پیش‌بینی مصرف انرژی در یک سیستم اینترنت اشیا ایجاد کنید.

مراحل انجام تمرین

۱. جمع آوری داده ها

- به یک دیتاست از مصرف انرژی دسترسی پیدا کنید. می‌توانید از دیتاست‌های عمومی مانند UCI Machine Learning Repository استفاده کنید.
- این دیتاست باید شامل ویژگی‌هایی مانند زمان، دما، نوع تجهیزات و مقدار انرژی مصرفی باشد.

۲. پیش‌پردازش داده‌ها

- داده‌ها را بررسی کنید و هر گونه داده گمشده یا نادرست را تصحیح کنید.
- ویژگی‌های غیرضروری را حذف کنید و در صورت نیاز داده‌ها را نرمال‌سازی کنید.

۳. ایجاد مدل یادگیری ماشین

- از یکی از کتابخانه‌های محبوب یادگیری ماشین مانند learn-scikit یا TensorFlowبرای ایجاد مدل استفاده کنید.
- از الگوریتم‌های مختلف مانند رگرسیون خطی، درخت تصمیم یا شبکه‌های عصبی استفاده کنید و نتایج آنها را مقایسه کنید.

٤. آزمایش و ارزیابی مدل

- مدل خود را با استفاده از داده‌های تست ارزیابی کنید. از معیارهایی مانند R^2، میانگین خطای مطلق(MAE) و میانگین مربعات خطا(MSE) استفاده کنید.
- نمودارهای مربوط به پیش‌بینی‌ها و مقادیر واقعی را رسم کنید تا عملکرد مدل خود را بررسی کنید.

۵. نتیجه گیری

- نتایج خود را تحلیل کنید و به این سوالات پاسخ دهید:
- کدام الگوریتم بهترین عملکرد را داشته است؟
- آیا ویژگی‌های خاصی بیشترین تأثیر را بر روی پیش‌بینی مصرف انرژی داشتند؟

ابزارها

- زبان برنامه نویسی: Python
- کتابخانه ها: Matplotlib، scikit-learn ،Pandas،NumPy

خروجی مورد انتظار

- یک گزارش شامل نتایج مدل‌های مختلف و تحلیل آنها
- کد منبع پروژه

```
1  # وارد کردن کتابخانه‌های مورد نیاز
2  import pandas as pd  # برای کار با داده‌های جدولی
3  import numpy as np  # برای انجام عملیات‌های عددی
4  from sklearn.model_selection import train_test_split  # برای تقسیم داده‌ها به مجموعه‌های آموزشی و تست
5  from sklearn.preprocessing import StandardScaler  # برای نرمال‌سازی داده‌ها
6  from sklearn.linear_model import LinearRegression  # مدل رگرسیون خطی
7  from sklearn.tree import DecisionTreeRegressor  # مدل درخت تصمیم
8  from sklearn.neural_network import MLPRegressor  # مدل شبکه عصبی (پرسیپترون چندلایه)
9  from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score  # معیارهای ارزیابی مدل
10 import matplotlib.pyplot as plt  # برای رسم نمودارها
11
```

کتاب خانه ها و ماژول ها

(pd) pandas

- برای کار با داده‌های جدولی به کار می‌رود. این کتابخانه امکان خواندن و نوشتن داده‌ها، عملیات‌های مختلف بر روی داده‌های جدولی (DataFrame)، فیلتر کردن، ترکیب، و مدیریت داده‌ها را فراهم می‌کند.

(np) numpy

- برای انجام عملیات‌های عددی و ریاضی کاربرد دارد. از آرایه‌های قدرتمند numpy برای انجام محاسبات سریع‌تر و کارآمدتر روی داده‌ها استفاده می‌شود.
- در این کد، numpy برای تبدیل سری داده‌ها به آرایه‌های عددی استفاده شده است.

(sklearn) scikit-learn

- این کتابخانه مجموعه‌ای از ابزارهای قدرتمند برای یادگیری ماشین فراهم می‌کند. در این کد از بخش‌های مختلف sklearn استفاده شده است.

- StandardScaler از sklearn.preprocessing: برای نرمال‌سازی و استانداردسازی داده‌ها به کار می‌رود.
- LinearRegression از sklearn.linear_model: برای ایجاد و آموزش مدل رگرسیون خطی استفاده می‌شود.
- DecisionTreeRegressor از sklearn.tree: برای ایجاد مدل درخت تصمیم به کار می‌رود.
- MLPRegressor از sklearn.neural_network: برای ایجاد مدل شبکه عصبی پرسپترون چندلایه (Multi-Layer Perceptron) استفاده شده است.
- mean_squared_error، mean_absolute_error، و r2_score از sklearn.metrics: این توابع برای محاسبه معیارهای ارزیابی مدل مانند خطای میانگین مربعات، خطای مطلق میانگین و ضریب تعیین (R^2) استفاده می‌شوند.

(plt) matplotlib.pyplot

- این کتابخانه برای رسم نمودارها و مصورسازی داده‌ها به کار می‌رود.

```
# مرحله ۱: جمع‌آوری داده‌ها
print("مرحله ۱: جمع‌آوری داده‌ها")

# مشخص کردن مسیر فایل داده‌ها
file_path = 'C:/Users/Mahdi/Desktop/household_power_consumption.txt'

# مشخص کردن جداکننده و نوع داده‌ها، خواندن داده‌ها از فایل .txt
data = pd.read_csv(
    file_path,
    delimiter=';',  # جداکننده برای داده‌های ورودی
    low_memory=False,  # برای خواندن فایل‌های بزرگ
    na_values='?',  # مقادیر نامعتبر را به NaN تبدیل می‌کند
    dtype={  # تعیین نوع داده‌ها برای هر ستون
        'Global_active_power': 'float64',
        'Global_reactive_power': 'float64',
        'Voltage': 'float64',
        'Global_intensity': 'float64',
        'Sub_metering_1': 'float64',
        'Sub_metering_2': 'float64',
        'Sub_metering_3': 'float64'
    }
)
print("داده‌ها با موفقیت جمع‌آوری شدند")
print(data.head())  # نمایش چند سطر ابتدایی داده‌ها

# ترکیب ستون‌های تاریخ و زمان به یک ستون datetime و تنظیم آن به عنوان index
data['DateTime'] = pd.to_datetime(data['Date'] + ' ' + data['Time'], format='%d/%m/%Y %H:%M:%S')
data = data.drop(columns=['Date', 'Time'])  # حذف ستون‌های اصلی تاریخ و زمان
data = data.set_index('DateTime')  # تنظیم ستون DateTime به عنوان شاخص
```

جمع آوری داده ها

در این کد، داده‌ها از یک فایل متنی با نام household_power_consumption.txtجمع‌آوری شده‌اند. فرآیند جمع‌آوری داده‌ها به این صورت انجام شده است:

مشخص کردن مسیر فایل

- مسیر فایل داده‌ها در متغیری به نام file_path ذخیره شده است.
- این فایل حاوی اطلاعات مصرف انرژی است.

خواندن داده‌ها از فایل

- داده‌ها با استفاده از دستور pd.read_csv() از فایل متنی خوانده شده‌اند. در اینجا چندین پارامتر برای مشخص کردن فرمت و شرایط داده‌ها تنظیم شده‌اند:
  - delimiter=';' : جداکننده بین ستون‌های داده‌ها ; در نظر گرفته شده است.
  - low_memory=False: این گزینه برای بهینه‌سازی و جلوگیری از خواندن داده‌های حجیم به صورت تدریجی و با حافظه کم استفاده شده است.
  - na_values='?': تمامی مقادیر گم شده که با ? مشخص شده‌اند، به عنوان NaN (مقادیر نامعتبر) در نظر گرفته می‌شوند.
  - dtype: نوع داده برای برخی ستون‌ها به صورت float64 تعیین شده تا داده‌ها به صورت اعداد اعشاری ذخیره شوند.

```
42  # مرحله ۲: پیش‌پردازش داده‌ها
43  print("\nمرحله ۲: پیش‌پردازش داده‌ها")
44  print("اطلاعات مربوط به داده‌ها:")
45  print(data.info())  # نمایش اطلاعات اولیه داده‌ها
46
47  # حذف سطرهایی که داده‌های گمشده دارند
48  data = data.dropna()
49
50  # تقسیم داده‌ها به ویژگی‌ها (X) و هدف (y)
51  X = data.drop(columns=['Global_active_power'])  # حذف ستون هدف از ویژگی‌ها
52  y = data['Global_active_power']  # هدف: مصرف انرژی فعال به صورت جهانی
53
54  # نرمال‌سازی داده‌ها
55  scaler = StandardScaler()  # ایجاد شیء استانداردسازی
56  X = scaler.fit_transform(X)  # اعمال استانداردسازی به داده‌ها
57  print("پیش‌پردازش داده‌ها کامل شد")
58
59  # تقسیم داده‌ها به مجموعه‌های آموزشی و تست
60  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

پیش پردازش داده ها

پیش‌پردازش داده‌ها در این کد شامل مراحل مختلفی برای آماده‌سازی داده‌ها قبل از استفاده در مدل‌های یادگیری ماشین است. این مراحل به‌طور کامل به شرح زیر هستند:

ترکیب ستون‌های تاریخ و زمان و تنظیم شاخص (Index)

- ستون‌های Date و Time با یکدیگر ترکیب می‌شوند و یک ستون جدید به نام DateTime ایجاد می‌شود.
- این ستون به نوع داده‌ای datetime تبدیل و سپس به‌عنوان شاخص برای DataFrame تنظیم می‌شود. این کار کمک می‌کند تا داده‌ها به‌صورت زمان‌بندی‌شده (time-series) در دسترس باشند.

حذف داده‌های گم‌شده

- سطرهایی که شامل داده‌های گم‌شده (NaN) هستند، از DataFrame حذف می‌شوند. این کار برای اطمینان از کیفیت داده‌ها انجام می‌شود، زیرا مقادیر گم‌شده می‌توانند در عملکرد مدل اختلال ایجاد کنند.

تقسیم داده‌ها به ویژگی‌ها (X) و هدف(y)

- ویژگی‌های (متغیرهای ورودی) برای مدل‌سازی با حذف ستون هدف (Global_active_power) از DataFrame جدا می‌شوند. ستون هدف به‌عنوان متغیر وابسته (مقدار پیش‌بینی‌شده) انتخاب می‌شود.

نرمال‌سازی داده‌ها (استانداردسازی)

- داده‌های ویژگی‌ها (X) به کمک StandardScaler از scikit-learn استانداردسازی می‌شوند. این فرآیند داده‌ها را طوری مقیاس‌بندی می‌کند که میانگین داده‌ها صفر و انحراف معیار آن‌ها یک باشد.
- نرمال‌سازی داده‌ها به مدل کمک می‌کند تا با سرعت بیشتری همگرا شود و مشکلات مربوط به تفاوت مقیاس‌ها بین ویژگی‌های مختلف را کاهش دهد.

```
61  # مرحله ۳: ایجاد مدل‌های یادگیری ماشین
62  print("\nمرحله ۳: ایجاد مدل‌های یادگیری ماشین")
63  # تعریف مدل‌های مختلف
64  models = {
65      'Linear Regression': LinearRegression(),  # مدل رگرسیون خطی
66      'Decision Tree': DecisionTreeRegressor(random_state=42),  # مدل درخت تصمیم
67      'Neural Network (MLP)': MLPRegressor(hidden_layer_sizes=(100,), max_iter=200,
    random_state=42)  # مدل شبکه عصبی
68  }
69
70  results = {}  # دیکشنری برای ذخیره نتایج مدل‌ها
71
72  for name, model in models.items():
73      print(f"\nآموزش مدل: {name}")
74      if name == 'Neural Network (MLP)':  # آموزش مدل شبکه عصبی با داده‌های دسته‌بندی شده
75          chunk_size = 1000
76          print("آموزش مدل شبکه عصبی با استفاده از پردازش داده‌ها در دسته‌های کوچک...")
77          for i in range(0, X_train.shape[0], chunk_size):
78              X_chunk = X_train[i:i + chunk_size]
79              y_chunk = y_train.iloc[i:i + chunk_size].to_numpy()  # تبدیل Series به numpy array
80              model.partial_fit(X_chunk, y_chunk.ravel())  # استفاده از ravel() برای تبدیل به آرایه ۱D
81      else:
82          model.fit(X_train, y_train)  # آموزش مدل‌های دیگر
83
84      predictions = model.predict(X_test)  # پیش‌بینی بر اساس داده‌های تست
```

ایجاد مدل یادگیری ماشین

ایجاد مدل‌های یادگیری ماشین در کد به سه مرحله اصلی تقسیم شده است: تعریف مدل‌ها، آموزش مدل‌ها، و پیش‌بینی و ارزیابی مدل‌ها. در ادامه هر کدام از این مراحل به تفصیل توضیح داده می‌شوند:

تعریف مدل‌ها

سه مدل مختلف برای پیش‌بینی مصرف انرژی تعریف شده‌اند:

- رگرسیون خطی (LinearRegression)
- درخت تصمیم (DecisionTreeRegressor)
- شبکه عصبی پرسپترون چندلایه (MLPRegressor)

این مدل‌ها در یک دیکشنری به نام models تعریف شده‌اند.

توضیحات مربوط به هر مدل

- LinearRegression: مدل رگرسیون خطی ساده که رابطه خطی بین ویژگی‌ها و متغیر هدف را پیش‌بینی می‌کند.
- DecisionTreeRegressor: مدل درخت تصمیم که با ساختن گره‌ها و برگ‌ها، داده‌ها را به بخش‌های مختلف تقسیم کرده و پیش‌بینی‌های هدف را انجام می‌دهد.
- MLPRegressor: مدل شبکه عصبی با استفاده از پرسپترون چندلایه. این مدل از لایه‌های مخفی استفاده می‌کند و می‌تواند روابط پیچیده غیرخطی را بیاموزد. اینجا یک لایه مخفی با ۱۰۰ نورون تعریف شده و max_iter=200 تعداد تکرارها (epochs) را تعیین می‌کند.

آموزش مدل‌ها

برای هر مدل، فرآیند آموزش شامل فراخوانی متد fit() بر روی داده‌های آموزشی است. در صورتی که مدل از نوع Neural Network (MLP) باشد، آموزش به صورت جزئی (partial) انجام می‌شود.

آموزش مدل‌های رگرسیون خطی و درخت تصمیم

- برای مدل‌های رگرسیون خطی و درخت تصمیم، آموزش به سادگی با دستور model.fit() انجام می‌شود.

آموزش مدل شبکه عصبی (MLPRegressor)

- برای MLPRegressor، داده‌ها به صورت دسته‌های کوچک (چانک‌ها) پردازش می‌شوند. این روش کمک می‌کند تا از حافظه بهینه استفاده شده و مدل به‌صورت تدریجی با داده‌های ورودی تنظیم شود.
- داده‌ها به دسته‌های کوچکتر با اندازه chunk_size تقسیم می‌شوند.
- partial_fit() برای آموزش مدل با داده‌های دسته‌بندی شده استفاده می‌شود. این متد امکان یادگیری تدریجی را فراهم می‌کند و به‌خصوص برای داده‌های بزرگ مفید است.

```
85  # مرحله ۴: آزمایش و ارزیابی مدل
86      print("\nمرحله ۴: آزمایش و ارزیابی مدل")
87      # محاسبه معیارهای ارزیابی مدل
88      mse = mean_squared_error(y_test, predictions)
89      mae = mean_absolute_error(y_test, predictions)
91      r2 = r2_score(y_test, predictions)
92      results[name] = {'MSE': mse, 'MAE': mae, 'R2': r2}
93      # رسم نمودار برای مقایسه مقادیر واقعی و پیش‌بینی شده
94      plt.figure(figsize=(10, 6))
95      plt.plot(range(len(y_test)), y_test, label='Real', alpha=0.6)
96      plt.plot(range(len(y_test)), predictions, label='Predicted', alpha=0.6)
97      plt.title(f'{name} Predictions vs Real Values')
98      plt.xlabel('Samples')
99      plt.ylabel('Energy Consumption (kWh)')
100     plt.legend()
101     plt.pause(0.1)
102 print("\nنتایج مدل‌ها:")
103 for name, metrics in results.items():
104     print(f'{name} - MSE: {metrics["MSE"]}, MAE: {metrics["MAE"]}, R²: {metrics["R2"]}')
```

آزمایش و ارزیابی مدل

آزمایش و ارزیابی مدل‌ها در این کد به منظور بررسی عملکرد هر مدل یادگیری ماشین بر اساس داده‌های تست انجام می‌شود. این مرحله شامل پیش‌بینی با داده‌های تست و ارزیابی نتایج پیش‌بینی شده با استفاده از معیارهای مختلف است. در ادامه فرآیند به‌طور کامل توضیح داده می‌شود:

پیش‌بینی مقادیر با داده‌های تست

- برای هر مدل، پس از آموزش با داده‌های آموزشی (X_train و y_train)، داده‌های تست (X_test) به مدل وارد می‌شوند تا مقادیر هدف پیش‌بینی شوند. این کار با استفاده از متد predict() انجام می‌شود:
  - predictions شامل مقادیر پیش‌بینی شده توسط مدل است.
  - X_test داده‌هایی هستند که مدل قبلاً آن‌ها را ندیده و هدف از این پیش‌بینی، ارزیابی میزان دقت مدل بر اساس داده‌های جدید و ناآشنا است.

محاسبه معیارهای ارزیابی عملکرد

برای ارزیابی کیفیت پیش‌بینی مدل‌ها، از سه معیار اصلی استفاده شده است:

- خطای میانگین مربعات (Mean Squared Error - MSE): میانگین مربع اختلافات بین مقادیر واقعی (y_test) و مقادیر پیش‌بینی شده (predictions) را محاسبه می‌کند. مقدار کمتر برای MSE نشان‌دهنده دقت بالاتر است.
- خطای مطلق میانگین (Mean Absolute Error - MAE): میانگین مقدار مطلق اختلافات بین مقادیر واقعی و پیش‌بینی شده را محاسبه می‌کند. MAE نشان‌دهنده میانگین خطای مطلق در پیش‌بینی‌ها است و مقدار کمتر به معنای دقت بالاتر مدل است.
- ضریب تعیین (R^2 Score): معیاری است که مقدار واریانس داده‌ها را که توسط مدل توضیح داده می‌شود، نشان می‌دهد. مقدار R^2 نزدیک به ۱ نشان‌دهنده عملکرد خوب مدل است و اگر مقدار آن به ۰ یا منفی نزدیک باشد، به این معنا است که مدل به‌خوبی نتایج را پیش‌بینی نکرده است.

ذخیره نتایج ارزیابی

- نتایج محاسبه‌شده برای هر مدل در یک دیکشنری به نام results ذخیره می‌شود.
- این دیکشنری شامل نام مدل به‌عنوان کلید و مقادیر معیارها به‌عنوان مقادیر ذخیره شده است. این امر کمک می‌کند تا نتایج هر مدل به‌صورت مرتب نگه‌داری شود.

رسم نمودار برای مقایسه مقادیر واقعی و پیش‌بینی شده

برای هر مدل، نموداری رسم می‌شود که مقادیر واقعی و مقادیر پیش‌بینی شده توسط مدل را مقایسه می‌کند. این کار به کمک Matplotlib انجام می‌شود:

- محور افقی (x) نمونه‌ها را نشان می‌دهد و محور عمودی (y) مقادیر مصرف انرژی (هدف) است.
- این نمودارها دیدگاه بصری مفیدی برای بررسی کیفیت پیش‌بینی‌های هر مدل فراهم می‌کنند و تفاوت‌های بین مقادیر واقعی و پیش‌بینی شده را نشان می‌دهند.

```
105 # مرحله ۵: نتیجه‌گیری
106 print("\nمرحله ۵: نتیجه‌گیری")
107 best_model = max(results, key=lambda x: results[x]['R2'])
108 print(f'بهترین مدل: {best_model}')
```

نتیجه گیری

انتخاب بهترین مدل

- ابتدا مدل‌ها مقادیر داده‌های تست را پیش‌بینی می‌کنند.
- نتایج پیش‌بینی‌ها با داده‌های واقعی مقایسه شده و سه معیار (MSE، MAE، و R^2) محاسبه می‌شوند.
- بهترین مدل در کد بر اساس معیار ضریب تعیین (R^2) انتخاب شده است، زیرا این معیار نشان می‌دهد که مدل تا چه حد توانسته است واریانس داده‌های هدف (مقادیر واقعی) را توضیح دهد. در ادامه توضیح می‌دهم که چرا این معیار به‌عنوان معیار اصلی انتخاب شده است و چه مزایایی دارد:

- R^2 نشان‌دهنده نسبتی است که در آن واریانس داده‌های هدف توسط مدل توضیح داده می‌شود. مقدار آن بین ۰ و ۱ (گاهی اوقات منفی نیز می‌تواند باشد) متغیر است:
- مقایسه مدل‌ها: R^2 به شما امکان می‌دهد به‌راحتی مدل‌ها را با یکدیگر مقایسه کنید. مدل با مقدار بالاتر R^2 توانسته است واریانس بیشتری از داده‌های هدف را توضیح دهد و از این رو مدل بهتری محسوب می‌شود.
- تفسیر آسان: مقدار R^2 به سادگی تفسیرپذیر است؛ برای مثال، اگر R^2 برابر با ۰.۸ باشد، به این معنا است که مدل ۸۰٪ واریانس داده‌های هدف را توضیح داده است.
- ارتباط با دقت پیش‌بینی: مدل با مقدار بالاتر R^2 معمولاً دقت پیش‌بینی بهتری دارد (اگرچه همیشه نباید به تنهایی بر این معیار متکی بود). مقادیر R^2 بالاتر نشان‌دهنده این است که مدل پیش‌بینی‌های بهتری ارائه می‌دهد و به‌خوبی می‌تواند داده‌های تست را پیش‌بینی کند.

چرا معیارهای دیگری انتخاب نشده‌اند؟

- خطای میانگین مربعات (MSE) و خطای مطلق میانگین (MAE) نشان‌دهنده مقدار خطاهای پیش‌بینی هستند، اما این معیارها نمی‌گویند که مدل چقدر از تغییرات داده‌های واقعی را توضیح داده است.
- R^2 در کنار این معیارها برای ارزیابی کلی مدل بسیار مفید است و می‌تواند دید کاملی نسبت به توانایی مدل در توضیح تغییرات داده‌های هدف فراهم کند.
- به عنوان مثال، مدل با کمترین مقدار MSE یا MAE ممکن است واریانس کمتری از داده‌ها را توضیح دهد. در چنین شرایطی، R^2 می‌تواند کمک کند تا بهترین مدل را از نظر توضیح‌دهی واریانس انتخاب کنید، و نه فقط از نظر مقدار خطا.

جمع‌بندی

R^2 به‌عنوان یک معیار انتخاب شده است زیرا به خوبی نشان می‌دهد که مدل چقدر می‌تواند واریانس داده‌ها را توضیح دهد و به سادگی قابل تفسیر و مقایسه بین مدل‌ها است. این معیار می‌تواند اطلاعات مفیدی در مورد دقت و کیفیت پیش‌بینی مدل فراهم کند و به تصمیم‌گیری در مورد انتخاب بهترین مدل کمک می‌کند.

بر اساس معیارهای ارزیابی، به ویژه ضریب تعیین (R^2)، الگوریتم شبکه عصبی (MLP Regressor) بهترین عملکرد را داشته است. این مدل توانست بیشترین درصد واریانس را توضیح دهد و دقت بالاتری در پیش‌بینی‌ها داشت.

تأثیر ویژگی‌های خاص بر روی پیش‌بینی مصرف انرژی

- Voltage (ولتاژ): تغییرات ولتاژ می‌تواند تأثیر زیادی بر مصرف انرژی داشته باشد.
- Global_intensity (شدت جریان): این ویژگی نشان‌دهنده شدت جریان برق است که می‌تواند با مصرف انرژی همبستگی بالایی داشته باشد.
- Sub_metering_1, Sub_metering_2, Sub_metering_3 (مصرف در زیرسیستم‌ها): این ویژگی‌ها مصرف انرژی در بخش‌های مختلف را نشان می‌دهند که می‌توانند تأثیر زیادی بر پیش‌بینی کل مصرف انرژی داشته باشند.

در نهایت، مدل شبکه عصبی بهترین عملکرد را داشته و می‌توان فرض کرد که ویژگی‌های ولتاژ، شدت جریان، و مصرف در زیرسیستم‌ها بیشترین تأثیر را بر روی پیش‌بینی مصرف انرژی داشته‌اند. این نتایج می‌تواند به بهبود مدل‌ها و انتخاب ویژگی‌های مهم در تحلیل‌های آینده کمک کند.