نام پروژه : یادگیری عمیق به زبان برنامه نویسی پایتون

استاد : امین دهقان

دانشجو : علیرضا خراسانی

تاریخ : ۱۴۰۴/۰۲

خلاصه پروژه :

- پروژه شامل چهار فصل می باشد. فصل اول توضیح مختصری راجب زبان برنامه نویسی پایتون می باشد .فصل دوم توضیح مختصری راجب یادگیری ماشین . فصل سوم توضیح مختصری راجب یادگیری عمیق . فصل چهارم الگوریتم های استفاده شده در پروژه

فصل اول زبان برنامه نویسی پایتون

فصل ۱ . ۱ : مقدمه‌ای بر برنامه نویسی

برنامه‌نویسی یعنی گفتن به کامپیوتر که دقیقاً چه کاری رو انجام بده.

پایتون زبان ساده و سطح بالاست که برای مبتدی‌ها عالیه.

نوشتن اولین برنامه:

```
Print("Hello, world!")
```

فصل ۱ . ۲ : متغیرها و انواع داده

هدف :

آشنایی با متغیرها و رشته‌ها(strings)

مثال‌ها:

```
name = "Ali"

print(f"Hello, {name.title()}!")  # Hello, Ali!
```

نکات کاربردی :

- رشته‌ها با " " یا ' ' تعریف می‌شن
- متدهای رشته:
  - title().حروف اول کلمات بزرگ می‌شن
  - upper(), .lower() تبدیل به حروف بزرگ یا کوچک

فصل ۱ . ۳ : لیست‌ها(Lists)

هدف :

- ساخت و استفاده از لیست‌ها

مثال:

```
fruits = ["apple", "banana", "cherry"]

print(fruits[0])  # apple
```

عملیات مهم :

- اضافه کردن(append(), .insert().
- حذف(del, .pop(), .remove()

فصل ۱ . ۴ : کار با لیست‌ها

هدف :

- پیمایش لیست با حلقهfor ، مرتب‌سازی وslicing

مثال:

```
for fruit in fruits:

    print(f"I like {fruit}")
```

ابزار ها :

- range(1,6) اعداد ۱ تا ۵
- لیست کامپری هنشن : [x**2 for x in range(10)]

فصل ۱ . ۵ : شرط‌ها(if statements)

هدف:

- تصمیم‌گیری با شرط‌ها

مثال :

```
age = 18

if age >= 18:

    print("You can vote.")

else:

    print("Too young!")
```

ابزار ها :

- عملگرها==, !=, >, <, >=, <=
- and, or, not

فصل ۱ . ۶ : دیکشنری‌ها(Dictionaries)

هدف :

- کار با ساختار کلید-مقدار

مثال :

```
person = {"name": "Ali", "age": 25}

print(person["name"])
```

عملیات :

- افزودن کلید person["city"] = "Tehran"
- حذف del person["age"]

فصل ۱ . ۷ : ورودی کاربر و حلقه while

هدف :

- گرفتن ورودی از کاربر و استفاده از while

مثال :

```
name = input("What's your name? ")

print(f"Hi {name}!")
```

حلقه:while

```
count = 1

while count <= 5:

            print(count)

            count += 1
```

فصل ۱ . ۸ : توابع(Functions)

هدف :

- ساخت و استفاده از توابع

مثال :

```
def greet_user(name):

       print(f"Hello, {name}!")

greet_user("Ali")
```

انواع پارامتر :

- پیش‌فرض def greet(name="guest")
- دلخواه *args, **kwargs

فصل ۱ . ۹ : کلاس ها (Classes)

هدف :

برنامه‌نویسی شیءگرا با کلاس‌ها و اشیا

```
class Dog:

    def __init__(self, name):

        self.name = name

    def bark(self):

        print(f"{self.name} says woof!")

my_dog = Dog("Rex")

my_dog.bark()
```

فصل ۱ . ۱۰ : مدیریت خطاها و فایل ها

هدف :

- خواندن/ نوشتن فایل‌ها و گرفتن خطاها

فایل :

```
with open("message.txt", "w") as file:
    file.write("Hello file!")
```

مدیریت خطا:

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Can't divide by zero.")
```

منبع :[1]

فصل دوم یادگیری ماشین

خلاصه آموزش ماشین لرنینگ با مثال های کاربردی

فصل ۲ . ۱ : یادگیری ماشین چیست؟

- فرآیندی است که کامپیوتر با استفاده از داده‌ها، بدون برنامه‌نویسی صریح، الگوهایی را یاد می‌گیرد.
- مثال :پیش‌بینی قیمت خانه با توجه به متراژ، محل، و تعداد اتاق‌ها.

فصل ۲ . ۲ : انواع یادگیری ماشین

یادگیری تحت نظارت(Supervised Learning)

- داده‌ها شامل ورودی‌ها (X) و خروجی‌ها (Y) هستند.
- مثال: تشخیص اینکه ایمیل اسپم است یا نه.(classification)
- مثال: پیش‌بینی قیمت ماشین.(regression)

یادگیری بدون نظارت(Unsupervised Learning)

- فقط ورودی وجود دارد، بدون برچسب خروجی.
- مثال: خوشه‌بندی مشتریان با الگوهای خرید مشابه.(clustering)
- مثال: کاهش ابعاد ویژگی‌ها برای ساده‌سازی داده.(PCA)

فصل ۲ . ۳ : الگوریتم‌های رایج

الگوریتم	کاربرد	مثال عملی
Linear Regression	رگرسیون	پیش‌بینی قیمت خانه
Logistic Regression	طبقه‌بندی	پیش‌بینی بیماری (بله/خیر)
Decision Tree	طبقه‌بندی/رگرسیون	تصمیم‌گیری درباره خرید بیمه
Random Forest	طبقه‌بندی/رگرسیون	بهبود دقت با ترکیب چند درخت
k-NN	طبقه‌بندی	تشخیص نوع گل از روی ویژگی‌ها
SVM	طبقه‌بندی	جداسازی تصاویر گربه و سگ
Neural Network	مسائل پیچیده	تشخیص چهره، گفتار، ترجمه خودکار

فصل ۲ . ۴ : ارزیابی مدل ها

- تقسیم داده‌ها به train/test
- استفاده از Cross-validation
- معیارها:
  - دقت(Accuracy)
  - Precision, Recall, F1-Score
  - برای رگرسیون (RMSE)

فصل ۲ . ۵ : مشکلات رایج

- (Overfitting) بیش‌برازش : مدل بیش از حد داده‌ها را حفظ می‌کند ⟶ دقت بالا روی داده‌های آموزش، ولی ضعیف روی داده جدید.
- (Underfitting)کم‌برازش :مدل خیلی ساده است و نمی‌تواند الگوها را یاد بگیرد.

فصل ۲ . ۶ : بهبود عملکرد مدل ها

- Feature Engineering : ساخت ویژگی‌های بهتر و مؤثر.
- Hyperparameter Tuning : تنظیم دستی مقادیر مثل learning rate یا max depth.
- Ensemble Methods : ترکیب چند مدل مثل.(Random Forest)

فصل ۲ . ۷ : ابزارهای کاربردی در Python

کاربرد	کتابخانه
الگوریتم‌های کلاسیک ML	scikit-learn
ساخت مدل‌های عمیق (Deep Learning)	TensorFlow / Keras
پردازش و تحلیل داده‌ها	Pandas / NumPy
رسم نمودار و دیداری‌سازی	Matplotlib / Seaborn

نتیجه گیری

یادگیری ماشین ابزاری بسیار قدرتمند برای حل مسائل پیچیده است. ترکیب داده مناسب + الگوریتم درست + ارزیابی دقیق، کلید موفقیت در پروژه‌های ML است.

منبع : [2]

فصل سوم یادگیری عمیق

یادگیری عمیق شاخه‌ای از یادگیری ماشین است که با استفاده از شبکه‌های عصبی عمیق (Deep Neural Networks)، قادر به مدل‌سازی روابط غیرخطی بسیار پیچیده میان ورودی و خروجی است.

برخلاف مدل‌های سنتی که اغلب به ویژگی‌سازی دستی متکی هستند، یادگیری عمیق با استفاده از چندین لایه‌ی پنهان (Hidden Layers) ویژگی‌ها را به‌صورت خودکار یاد می‌گیرد.

فصل ۳ . ۱ : ساختار یک شبکه عصبی عمیق

ساختار شبکه شامل:

- لایه ورودی :(Input Layer) دریافت داده خام (مثلاً پیکسل‌های یک تصویر)
- لایه‌های پنهان :(Hidden Layers) تبدیل داده به ویژگی‌های سطح بالا (مثلاً خطوط، منحنی‌ها)
- لایه خروجی :(Output Layer) تولید پیش‌بینی نهایی (مثلاً تشخیص عدد ۷)

فصل ۳ . ۲ : فرایند آموزش

۱. پیش‌پردازش داده‌ها (نرمال‌سازی، تبدیل، تقسیم)

۲. انتقال داده از لایه‌ها

۳. محاسبه خطا(Loss)

٤. پس انتشار خطا (Backpropagation) و تنظیم وزن‌ها با کمک گرادیان نزولی (Gradient Descent)

فصل ۳ . ۳ : مزیت های یادگیری عمیق

مزیت	توضیح
یادگیری خودکار ویژگی‌ها	نیازی به استخراج دستی ویژگی نیست
توانایی مدل‌سازی پیچیدگی بالا	مناسب برای تصاویر، متن، گفتار و بازی‌ها
مقیاس‌پذیری بالا	می‌تواند با داده‌های بزرگ وGPU ها به خوبی کار کند

فصل ۳ . ۴ : چالش ها

۱. نیاز به داده زیاد

۲. مصرف محاسبات بالا

۳. نیاز به تنظیمات دقیق (Hyperparameters)

فصل ۳ . ۵ : یادگیری عمیق در چه جا هایی استفاده میشود ؟

- **بینایی ماشین :**تشخیص چهره، خودرو، اشیاء
- **پردازش زبان طبیعی :**ترجمه ماشینی، چت‌بات
- **گیمینگ :**هوش مصنوعی بازی‌ها
- **پزشکی :**تشخیص بیماری از تصاویر پزشکی
- **مالی :**پیش‌بینی بازار و تشخیص تقلب

منبع کتاب : [3]

فصل چهارم الگوریتم های استفاده شده در پروژه

دیتاست استفاده شده در پروژه Parkinson's نام دارد

دانلود شده از سایت UCI Machin Learning Repository

مجموعه داده‌های تشخیص بیماری پارکینسون آکسفورد

ویژگی ها	نمونه ها	نوع ویژگی	وظایف مرتبط	حوزه موضوعی	ویژگی مجموعه داده ها
۲۲	۱۹۷	واقعی	طبقه بندی	سلامت و پزشکی	چند متغیره

فصل ۴ . ۱ : اطلاعات مجموعه داده ها

این مجموعه داده شامل طیف وسیعی از اندازه‌گیری‌های صدای زیست‌پزشکی از ۳۱ نفر است که ۲۳ نفر از آنها مبتلا به بیماری پارکینسون هستند. هر ستون در جدول یک معیار صدای خاص است و هر ردیف مربوط به یکی از ۱۹۵ صدای ضبط شده از این افراد (ستون "نام") است. هدف اصلی داده‌ها، تمایز افراد سالم از افراد مبتلا به پارکینسون، بر اساس ستون "وضعیت" است که برای افراد سالم روی ۰ و برای افراد مبتلا به پارکینسون روی ۱ تنظیم شده است.

فصل ۴ . ۲ : *الگوریتم ها*

- Random Forest
- Svm(Support Vector Machine)
- Stacking
- Bagging
- Boosting
- Knn(K-Nearest Neighbors)
- Decision Tree

فصل ۴ . ۳ : توضیح کوتاه در مورد الگوریتم ها

الگوریتم رندوم فورست (Random Forest) یک مدل یادگیری ماشین است که از ترکیب چندین درخت تصمیم (Decision Trees) استفاده می‌کند. هر درخت روی زیرمجموعه‌ای تصادفی از داده‌ها و ویژگی‌ها آموزش می‌بیند، و در نهایت میانگین یا رای‌گیری نتایج درخت‌ها به عنوان خروجی نهایی استفاده می‌شود.

مزیت :دقت بالا و کاهش بیش‌برازش.(Overfitting)

کاربرد :طبقه‌بندی و رگرسیون.

SVM (Support Vector Machin):

الگوریتمی برای طبقه‌بندی که بهترین خط یا مرز را پیدا می‌کند تا داده‌ها را با بیشترین فاصله از هم جدا کند.

Stacking:

ترکیب چند مدل مختلف (مثل درخت، SVMو KNN) و استفاده از

مدل نهایی (meta-model) برای تصمیم‌گیری نهایی.

Bagging (Bootstrap Aggregating):

ساخت چندین مدل مشابه روی نمونه‌گیری تصادفی از داده‌ها و ترکیب نتایج آن‌ها (مثلاً Random Forestنوعی Bagging است).

Boosting :

مدل‌ها به صورت زنجیره‌ای ساخته می‌شوند؛ هر مدل سعی می‌کند خطاهای مدل قبلی را اصلاح کند (مثل.XGBoost)

KNN (K-Nearest Neighbors) :

داده جدید را بر اساس نزدیکی به k نمونه‌ی قبلی طبقه‌بندی یا پیش‌بینی می‌کند.

Decision Tree:

الگوریتمی برای طبقه‌بندی یا پیش‌بینی است که داده‌ها را با پرسیدن سوال‌های بله/خیر (یا مقایسه‌ای) به صورت درختی تقسیم می‌کند.

فصل ۴. ۴ : Accuracy / Confusion Matrix چیست ؟

	مقدار های واقعی		
		+	**-**
مقدار های پیش بینی شده	**+**	TP	FP
	-	FN	TN

توضیحات جدول:

- TP : True Positive
  - مقدار واقعی + بوده و الگوریتم هم + پیش بینی کرده است.
- FP : False Positive
  - مقدار واقعی - بوده اما الگوریتم + پیش بینی کرد.
- FN : False Negative
  - مقدار واقعی + بوده و الگوریتم - پیش بینی کرده

- TN : True Negative

  - مقدار واقعی - بوده الگوریتم هم - پیش بینی کرده

روش بدست آوردن Accuracy :

فرمول : $\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$

فرض می کنیم یک ماتریس ۲ * ۲ داریم

$$\begin{bmatrix} 30 & 2 \\ 1 & 40 \end{bmatrix}$$

روش بدست آوردن دقت ماتریس بالا:

$$\text{Accuracy} = \frac{30+40}{30+2+40+1} = 0.95$$

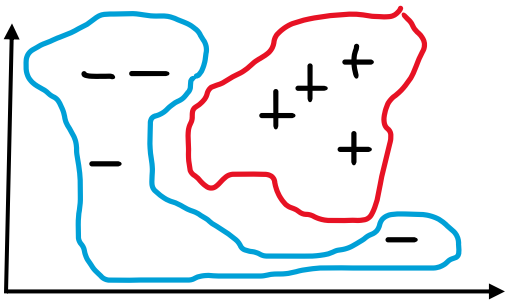
فصل ۴ . ۵ : *accuracy_score* هر الگوریتم استفاده شده در پروژه به شرح زیر می باشد

Base Lins	
Random Forest	1
Decision Trre	0,92
Knn	0,94
svm	0,98
Random Forest (smote)	0,89

Ensemble	
Bagging(Decision Tree=50)	0,94
Bagging	0,87
Stacking	0,94
Boosting	0,92
Voting(soft)	0,92
Voting(default (baseline knn,Rf,dt,svc)	0,94
Stacking(Only with base line Rf and knn)	0,94

در الگوریتم Random Forest مشکلی که وجود داشت Overfittting رخ داد . یعنی بیش برازش احتمالا به دلیل اینکه دیتاست کوچک بود.

مانند شکل زیر رخ داده است.



بعد از اینکه Over fittting رخ داد SMOTE به دیتاست اضافه کردیم

Random _state یعنی مولد اعداد تصادفی است

smote.fit _ resample یعنی نمونه گیری مجدد از مجموعه داده

در نتیجه accuracy _score شد 0,89

فصل ۴ . ۶ : منابع

[1] : Python Crash Course – A Hands-On, Project-Based Introduction to Programming

Effect : Eric Matthes

[2]: Hands-On Machine Learning

Effect : Aurelien Geron

[3] : Deep Learning

Written by: Ian Goodfellow, Yoshua Bengio, and Aaron Courville

Publications : MIT Press