



**T.C.
NEVŞEHİR HACI BEKTAŞ VELİ
ÜNİVERSİTESİ
MÜHENDİSLİK-MİMARLIK FAKÜLTESİ**



STAJ DEFTERİ

ÖĞRENCİNİN	
Adı Soyadı:	Büşra ÇİLEK
Numarası :	20260810033
Bölümü	: Bilgisayar Mühendisliği

22/07/2024

T.C.
NEVŞEHİR HACI BEKTAŞ VELİ ÜNİVERSİTESİ
MÜHENDİSLİK-MİMARLIK FAKÜLTESİ

PRATİK ÇALIŞMA DEFTERİ

ÖĞRENCİNİN	Bölümü	Bilgisayar Mühendisliği	
	Numarası	20260810033	
	Adı-Soyadı	Büşra ÇİLEK	
	Fakülteye Girdiği Yıl	2021	
	Çalışma devresi	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3	
	Pratik Çalışmanın Başladığı Tarih	22/07/2024	
	Pratik Çalışmanın Bittiği Tarih	06/09/2024	
	Kaç İş Günü Pratik Çalışma Yaptığı	(30) iş günü	

İŞYERİNİN	İsim ve Adresi: Etiya Bilgi Teknolojileri Yazılım Sanayi Ticaret A.Ş Çifte Havuzlar Mahallesi Eski Londra Asfaltı Caddesi A2 Blok 151/1B No:101 YTÜ Davutpaşa Kampüsü Teknopark Esenler-İstanbul	
	İşyeri Adına Defteri Onaylayan Amirin:	Yukarıda kimliği ve fotoğrafı bulunan öğrencinin işyerimizde 30 iş günü pratik çalışma yaptığını ve bu defteri kendisinin düzenlediğini onaylarım. 06/09/2024
	Adı : Elif	
	Soyadı : AKGÜN	
	Ünvanı : Bilgisayar Mühendisi	
	Dip. No. : 140790	
	Oda Sicil No : 6659(BMO)	

Yapılan Çalışma:	<input type="checkbox"/>	
.....iş günlükdevre çalışma olarak kabul edilmiştir.	<input type="checkbox"/>	
Kabule uygun görülmemiştir.	<input type="checkbox"/>	
Komisyon Başkanı	Üye	Üye
Adı ve Soyadı:		
İmza :		

Tarih	:/...../20..../...../20..../...../20....
-------	---	--------------------	--------------------	--------------------

STAJ RAPORU ÇALIŞMA TAKVİMİ SAYFASI

22/07/2024 Tarihinden 06/09/2024 Tarihine Kadar Yapılan Toplam 30 Gün Çalışma		
Tarih	Yapılan İşler	Sayfa No
22/07/2024	Staj rehberi belirlendi, verilecek proje alanı belirlendi.	1
23/07/2024	Daily toplantısına girildi ve proje konusu araştırıldı.	2
24/07/2024	Proje konusu belirledim, rehberim proje konusu verdi.	3
25/07/2024	Projeye başladım, kurulum ve çalıştırma kısmına geçildi.	4
26/07/2024	Rastgele rakamlar vererek çıkışlara göre yanıt veren basit bir algoritma oluşturdum.	5
29/07/2024	Algoritmaya haftalık ortalama alarak çıkış verecek kodu ekledim.	6
30/07/2024	MSE ve RMSE hesaplamaları ekledim.	7
31/07/2024	Dosya yapısı oluşturuldu, Flask çalıştırıldı.	8
01/08/2024	Üretim ortamında Unicorn ile çalıştırmak için araştırmalar yapıldı.	9
02/08/2024	Windows için uygun olan waitress ile üretim ortamında çalıştırıldı.	10
05/08/2024	Flask uygulamasının ana dosyası oluşturuldu.	11
06/08/2024	API isteğini alması için predict Endpoint oluşturuldu.	12
07/08/2024	Yapay Zeka modelini içeren dosya oluşturuldu.	13
08/08/2024	Yapay Zeka modeli oluturuldu.	14
09/08/2024	Postmanda örnek istek ve yanıt denendi ve hata alındı.	15
12/08/2024	Kontroller yapıldı ve doğru sonuç alındı.	16
13/08/2024	Yanıt içerikleri incelendi ve doğrulandı.	17
14/08/2024	Hata Yönetimi (Error Handling) eklendi. Etiya stajyerlerle CV nasıl hazırlanır toplantısına katılındı.	18
15/08/2024	Login Rotası (Login Route) oluşturuldu.	19
16/08/2024	JWT ile güvenlik sistemi oluşturulmaya başlandı.	20
19/08/2024	Korumalı bir rotaya erişmek için bir JWT (JSON Web Token) doğrulaması yapıldı ve erişildi.	21
20/08/2024	Yapay Zeka Modeli Geliştirme ve Kullanma Adımları incelendi.	22
21/08/2024	Veri ön işleme adımı tamamlandı.	23
22/08/2024	Model geliştirme adımı tamamlandı.	24
23/08/2024	Flask API'de Modeli Kullanma adımına başlandı.	25
02/09/2024	Flask API'de Modeli Kullanma araştırmaları yapıldı.	26
03/09/2024	Flask API'de Modeli Kullanma kodları tamamlandı.	27
04/09/2024	Model eğitimi başarıyla tamamlandı.	28
05/09/2024	Örnek istek ve yanıt denendi ve doğrulandı.	29
06/09/2024	Proje sunumu yapıldı ve onaylandı.	30

Etiya Bilgi Teknolojileri Yazılım San ve Tic Şirketi	Sayfa No: 0
Rapor Yazım Kuralları 1. Staj raporları, Yönerge Eki ve Bölüm Staj Kılavuzlarında tanımlanan formata uygun biçimde hazırlanır. Bu koşullara uygun olmayan raporlar değerlendirmeye alınmaz ve öğrencinin stajı başarısız sayılır. 2. Raporlar A4 boyutunda beyaz kağıtlara, belirtilen şablona uygun bir şekilde, 12 pt Times New Roman fontları kullanılarak, tek satır aralığında (paragraf içleri) yazılır. Paragraflar arasında 1 satır boşluğu bırakılmalıdır. 3. Rapor bölümleri Latin karakterleri kullanılarak sayısal biçimde numaralanır (1., 2., 3. gibi). Alt bölümler de benzer biçimde numaralandırılır (1.1, 1.2, 1.3 gibi). Tüm bölümlere başlık konur (Alt bölümler dahil). Ana bölüm başlıkları büyük harflerle yazılır. Alt bölümlerde en çok 3 seviyeye kadar inilir (2.1.1 gibi). Bundan sonraki alt bölümler, eğer gerekirse küçük harf kullanılarak belirtilir (a), b), c) gibi). 4. Yazılan staj raporlarında örneği verilen kapak sayfası ve iç kapak sayfası(fotoğraflı) ile Çalışma Takvimi yer almalıdır. Söz konusu rapor, Bölüm tarafından belirlenen içeriğe uygun şekilde hazırlanır. 5. Yazılan rapora, gerekli görüldüğü takdirde ekler de konulabilir. Söz konusu ekler, EK-1, EK-2 biçiminde ayrılır ve bu eklerin, eğer varsa, bölüm numaraları E.1, E.2 biçiminde numaralanır. Rapora ve eklerine gereksiz hiçbir bilgi ve belge konulmaz. 6. Staj raporu Bölüm Staj Komisyonunun istediği şekilde bilgisayar ortamında yazılmış ve ciltlenmiş şekilde ilgili bölüme sunulur.	
Çalışma Tarihi:/...../20...	Onaylayanın, Adı – Soyadı: İmzası:.....

Etiya Bilgi Teknolojileri Yazılım San ve Tic A.Ş	Sayfa No: 1
<p>Stajımın ilk günü, stajyerler gruplar halinde mentorlarına atandı. Bana mentor olarak, Akbank Yapay Zeka ve Açık Bankacılık biriminde görev yapan Elif AKGÜN rehberlik etmektedir. Yapay zeka alanında çalışma isteğimi dile getirdiğim için bu alanda bir proje geliştirme kararı aldım. İlk gün, yapmak istediğim proje ve çalışmak istediğimiz alan ile ilgili bir toplantı gerçekleştirdik. Bu toplantının ardından, bir hafta içerisinde proje konusunu belirlememiz istendi. Toplantının sonrasında, şirketin genel yapısı ve faaliyetleri hakkında detaylı bilgilendirme yapıldı.</p>	
Çalışma Tarihi: 22/07/2024	Onaylayanın, Adı – Soyadı: Elif AKGÜN İmzası:.....

Etiya Bilgi Teknolojileri Yazılım San ve Tic A.Ş	Sayfa No: 2
<p>Stajımın ikinci gününde, ilk olarak Elif Hanım Akbanka bağlı olarak çalıştığı için bizi akbank toplantılarına girmemizi ve işleyişi öğrenmemizi istedi bugün sadece daily toplantısı vardı ve dün ne yaptım bugün ne yapacağım hakkında bilgilendirme yaptık. Sonrasında ise yapay zeka alanında gerçekleştirmek istediğim proje için konu araştırmaları yaptım. Bu alandada, alanda daha önce yapılmış projeleri ve bunlara ait kodları inceledim. Ayrıca, yapay zeka hakkında bilgi toplama ve derinlemesine araştırmalar gerçekleştirdim. Kaggle platformu (Kaggle, veri bilimi ve makine öğrenimi alanında çalışan veya bu alanlara ilgi duyan kişilerin bilgi ve yeteneklerini geliştirebileceği, projeler yapabileceği ve diğer uzmanlarla etkileşim kurabileceği bir platform) üzerinden çeşitli veri setlerini inceleyerek, projemin başlangıç aşamalarını ve nasıl ilerlemem gerektiğini planladım. Bu süreçte, ilgili kursları araştırarak, proje geliştirme sürecinde nasıl bir yol izlemem gerektiğini belirlemeye çalıştım. Son olarak, birkaç potansiyel proje konusu belirleyip, bunların nasıl gerçekleştirilebileceği üzerine detaylı araştırmalar yaptım.</p>	
Çalışma Tarihi: 23/07/2024	Onaylayanın, Adı – Soyadı: Elif AKGÜN İmzası:.....

Etia Bilgi Teknolojileri Yazılım San ve Tic A.Ş	Sayfa No: 3
<p>Stajımın üçüncü gününde, yine Akbank'ın daily toplantısına katıldım bu toplantıda dün ne yaptım bugün ne yapacağım hakkında bilgi veriyoruz ve bu her gün devam edecek. Daha önce belirlediğim konular arasından birine karar verdim ve bu konuyu mentoruma ilettim. Projemi, film veya dizi platformlarında kullanıcıların izlediği türlere göre öneriler sunan bir yapay zeka uygulaması olarak seçtim. Mentorüm, diğer bir stajyerle uyumlu çalışabileceğimiz bir öneride bulundu. Ancak, alanlarımızın farklı olmasından dolayı, birbiriyle bağlantılı ancak ayrı iki proje üzerinde çalışabileceğimizi belirtti. Mentorümün önerisi üzerine, diğer stajyer bir banka uygulaması geliştirecek, ben ise bu banka uygulamasında hesap giriş-çıkışlarını izleyerek anormal durumlarda geri bildirim verebilecek bir yapay zeka uygulaması yapacağım. Konuyu netleştirdikten sonra bu konu üzerine detaylı araştırmalar yaptım.</p>	
Çalışma Tarihi: 24/07/2024	Onaylayanın, Adı – Soyadı: Elif AKGÜN İmzası:.....

Projeme başladım.

1. Proje Tanıtımı:

Bu proje, bir bankacılık uygulamasında hesap hareketlerine dayalı öngörüler sağlayan basit bir yapay zeka modülüdür. Python ve Flask kullanılarak geliştirilecek ve REST API olarak sunulacaktır.

2. Teknolojiler:

- Python: Yapay zeka modelleri ve API için ana dil.
- Flask: Web sunucusu ve API yönetimi için.
- Scikit-learn / TensorFlow: Yapay zeka modelleri için (ihtiyaca göre seçilebilir).
- Pandas: Veri işleme ve analizi için.
- Numpy: Sayısal hesaplamalar için.
- Gunicorn: Üretim ortamında Flask uygulamasını çalıştırmak için.

3. Kurulum ve Çalıştırma

A. Gerekli Kütüphanelerin Kurulumu

Aşağıdaki komut ile gerekli Python kütüphanelerini kurun:

bash

pip install flask pandas numpy scikit-learn gunicorn

```
In [1]: pip install flask pandas numpy scikit-learn gunicorn
Requirement already satisfied: flask in c:\users\ymö\anaconda333\lib\site-packages (2.2.5)
Requirement already satisfied: pandas in c:\users\ymö\anaconda333\lib\site-packages (2.1.4)
Requirement already satisfied: numpy in c:\users\ymö\anaconda333\lib\site-packages (1.26.4)
Requirement already satisfied: scikit-learn in c:\users\ymö\anaconda333\lib\site-packages (1.2.2)
Requirement already satisfied: gunicorn in c:\users\ymö\anaconda333\lib\site-packages (23.0.0)
Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\ymö\anaconda333\lib\site-packages (from flask) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in c:\users\ymö\anaconda333\lib\site-packages (from flask) (3.1.3)
Requirement already satisfied: itsdangerous>=2.0 in c:\users\ymö\anaconda333\lib\site-packages (from flask) (2.0.1)
Requirement already satisfied: click>=8.0 in c:\users\ymö\anaconda333\lib\site-packages (from flask) (8.1.7)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\ymö\anaconda333\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\ymö\anaconda333\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\ymö\anaconda333\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.3.2 in c:\users\ymö\anaconda333\lib\site-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in c:\users\ymö\anaconda333\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ymö\anaconda333\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: packaging in c:\users\ymö\anaconda333\lib\site-packages (from gunicorn) (23.1)
Requirement already satisfied: colorama in c:\users\ymö\anaconda333\lib\site-packages (from click>=8.0->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\ymö\anaconda333\lib\site-packages (from Jinja2>=3.0->flask) (2.1.3)
Requirement already satisfied: six>=1.5 in c:\users\ymö\anaconda333\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Çalışma Tarihi: 25/07/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

```
1 import pandas as pd
2 from sklearn.ensemble import IsolationForest
3 data = {
4     'transaction_amount': [100, 200, 300, 4000, 150, 250, 300, 350, 5000, 1000],
5     'transaction_type': ['deposit', 'withdrawal', 'deposit', 'withdrawal', 'deposit', 'withdrawal', 'deposit', 'withdrawal', 'withdrawal', 'deposit']
6 }
7 df = pd.DataFrame(data)
8 df['transaction_amount'] = df['transaction_amount'].astype(float)
9 features = df[['transaction_amount']]
10 model = IsolationForest(contamination=0.2)
11 model.fit(features)
12
13 df['anomaly'] = model.predict(features)
14 df['is_risky'] = df['anomaly'] == -1
15
16 print(df)
```

	transaction_amount	transaction_type	anomaly	is_risky
0	100.0	deposit	1	False
1	200.0	withdrawal	1	False
2	300.0	deposit	1	False
3	4000.0	withdrawal	-1	True
4	150.0	deposit	1	False
5	250.0	withdrawal	1	False
6	300.0	deposit	1	False
7	350.0	withdrawal	1	False
8	5000.0	withdrawal	-1	True
9	1000.0	deposit	1	False

Rastgele rakamlar vererek hesaba giriş çıkış yapan sayıların normalden farklı olduğunda 1 ve true normal bir durumda ise -1 ve false sonuç verecek şekilde basit bir algoritma oluşturdum.

“transaction_amount” sütunu işlemlerin miktarını, “transaction_type” sütunu ise işlemin türünü (para yatırma veya çekme) temsil ediyor.

“transaction_amount” sütunu, sayısal değerlere (float) dönüştürülüyor. Bu adım, verilerin modelde kullanılabilmesi için gerekli.

“IsolationForest” adlı bir anormali tespit modeli oluşturuluyor. Bu model, veri setindeki "anormal" işlemleri tespit etmek için kullanılıyor. contamination=0.2 parametresi, veri setinde anormal olarak kabul edilecek işlemlerin oranını belirliyor.

Çalışma Tarihi: 26/07/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

```
1 import pandas as pd
2 from sklearn.ensemble import IsolationForest
3 data = {
4     'transaction_amount': [100, 200, 300, 4000, 150, 250, 300, 350, 5000, 1000],
5     'transaction_type': ['deposit', 'withdrawal', 'deposit', 'withdrawal', 'deposit', 'withdrawal', 'deposit', 'withdrawal', 'withdrawal', 'deposit'],
6     'transaction_date': [
7         '2023-07-01', '2023-07-02', '2023-07-03', '2023-07-04', '2023-07-05',
8         '2023-07-06', '2023-07-07', '2023-07-08', '2023-07-09', '2023-07-10'
9     ]
10 }
11 df = pd.DataFrame(data)
12 df['transaction_date'] = pd.to_datetime(df['transaction_date'])
13 df.set_index('transaction_date', inplace=True)
14 weekly_avg = df.resample('W')['transaction_amount'].mean().reset_index()
15 weekly_avg['transaction_amount'] = weekly_avg['transaction_amount'].astype(float)
16 features = weekly_avg[['transaction_amount']]
17
18 model = IsolationForest(contamination=0.2)
19 model.fit(features)
20 weekly_avg['anomaly'] = model.predict(features)
21
22 weekly_avg['is_risky'] = weekly_avg['anomaly'] == -1
23
24 print(weekly_avg)
25
```

	transaction_date	transaction_amount	anomaly	is_risky
0	2023-07-02	150.000000	-1	True
1	2023-07-09	1478.571429	1	False
2	2023-07-16	1000.000000	1	False

Burada yapılan hesap giriş çıkışlarının haftalık hesap giriş çıkışlarının ortalamasını alıp normal durumlarda -1 ve true, anormal durumlarda ise 1 ve false çıkışı veren basit bir algoritma oluşturdum.

“transaction_date” sütunundaki tarihler, pandas’ın “to_datetime” fonksiyonu kullanılarak datetime formatına dönüştürülüyor.


“resample” fonksiyonu ile veri haftalık olarak yeniden örnekleniyor. Yani, her bir haftanın ortalama işlem miktarı hesaplanıyor.

“reset_index” ile bu haftalık ortalama değerlerin tekrar bir DataFrame olarak düzenlenmesi sağlanıyor.

Çalışma Tarihi: 29/07/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

```
12 df = pd.DataFrame(data)
13 df['transaction_date'] = pd.to_datetime(df['transaction_date'])
14 daily_features = df[['transaction_amount']]
15 daily_model = IsolationForest(contamination=0.2, random_state=42)
16 daily_model.fit(daily_features)
17
18 df['daily_anomaly'] = daily_model.predict(daily_features)
19 df['is_daily_risky'] = df['daily_anomaly'] == -1
20
21 weekly_avg = df.resample('W-Mon', on='transaction_date')['transaction_amount'].mean().reset_index()
22 weekly_features = weekly_avg[['transaction_amount']]
23 weekly_model = IsolationForest(contamination=0.2, random_state=42)
24 weekly_model.fit(weekly_features)
25
26 weekly_avg['weekly_anomaly'] = weekly_model.predict(weekly_features)
27 weekly_avg['is_weekly_risky'] = weekly_avg['weekly_anomaly'] == -1
28
29 true_daily_anomalies = [0, 0, 0, 1, 0, 0, 0, 0, 1, 0]
30 true_weekly_anomalies = [1, 0]
31
32 daily_mse = mean_squared_error(true_daily_anomalies, df['is_daily_risky'].astype(int))
33 daily_rmse = np.sqrt(daily_mse)
34
35 weekly_mse = mean_squared_error(true_weekly_anomalies, weekly_avg['is_weekly_risky'].astype(int))
36 weekly_rmse = np.sqrt(weekly_mse)
37
38 print("Günlük Analiz Sonuçları:")
39 print(df[['transaction_date', 'transaction_amount', 'is_daily_risky']])
40 print("Günlük MSE: (daily_mse)")
41 print("Günlük RMSE: (daily_rmse)")
42
43 print("Haftalık Analiz Sonuçları:")
44 print(weekly_avg[['transaction_date', 'transaction_amount', 'is_weekly_risky']])
45 print("Haftalık MSE: (weekly_mse)")
46 print("Haftalık RMSE: (weekly_rmse)")
```



Günlük Analiz Sonuçları:

	transaction_date	transaction_amount	is_daily_risky
0	2023-07-01	100	False
1	2023-07-02	200	False
2	2023-07-03	300	False
3	2023-07-04	4000	True
4	2023-07-05	150	False
5	2023-07-06	250	False
6	2023-07-07	300	False
7	2023-07-08	350	False
8	2023-07-09	5000	True
9	2023-07-10	1000	False

Günlük MSE: 0.0
Günlük RMSE: 0.0

Haftalık Analiz Sonuçları:

	transaction_date	transaction_amount	is_weekly_risky
0	2023-07-03	200.000000	False
1	2023-07-10	1578.571429	False

Haftalık MSE: 0.5
Haftalık RMSE: 0.7071067811865476

Burada diğer güne ek olarak hata hesaplaması ekledim MSE ve RMSE hesaplamaları yapıyor. Günlük ve haftalık olarak MSE ve RMSE hesaplamalarını ayrı olarak yapıyor İlk olarak, işlem verileri bir pandas DataFrame'e aktarılır ve tarih sütunu datetime formatına dönüştürülür. Günlük işlem miktarları üzerinden model eğitilir ve her işlem için anormal olup olmadığı belirlenir. Daha sonra, haftalık ortalama işlem miktarları hesaplanır ve bu verilerle ikinci bir model eğitilerek haftalık anomali tespiti yapılır. Kod ayrıca, günlük ve haftalık anomali tespitlerinin doğruluğunu değerlendirmek için Mean Squared Error (MSE) ve Root Mean Squared Error (RMSE) metriklerini hesaplar. Sonuç olarak, hem günlük hem de haftalık bazda hangi işlemlerin riskli olduğu ve bu tespitlerin doğruluk oranları ekrana yazdırılır.

MSE, tahmin edilen değerler ile gerçek değerler arasındaki farkların karesinin ortalamasıdır.

Formül: $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

- n: Veri setindeki toplam örnek sayısı
- y_i : Gerçek değer
- \hat{y}_i : Tahmin edilen değer

MSE, hataların karesini alarak, pozitif ve negatif hataların birbirini dengelemesini önler. Hataların karesi alındığı için, büyük hatalar MSE üzerinde daha fazla etkiye sahiptir.

RMSE, MSE'nin karekökü alınarak hesaplanır ve sonuçlar orijinal birimlerde ifade edilir. Formül: $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$ (karekök eklenmiş hali)

RMSE, MSE'ye göre daha sezgiseldir çünkü hatalar orijinal ölçekte ifade edilir. Yüksek RMSE, modelin tahminlerinde büyük hatalar olduğunu göster

Çalışma Tarihi: 30/07/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

```
1 !ls -R project-root

project-root:
app.py  model.py  requirements.txt  templates

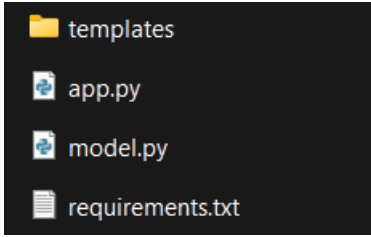
project-root/templates:
index.html
```

Benden istenilen dosya yapısı:

B. Proje Dosya Yapısı

plaintext project-root/

```
|— app.py
|— model.py
|— requirements.txt
|— templates/
    |— index.html
```



C. Çalıştırma

Flask uygulamasını başlatmak için:

bash

export FLASK_APP=app.py

flask run

```
In [2]: runfile('C:/Users/ymö/Desktop/project-r
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not u
instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
```

Çalışma Tarihi: 31/07/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

Flask Nedir?

Python programlama dili ile yazılmış, hafif ve esnek bir web geliştirme framework'üdür. Minimal bir yapı sunarak, geliştiricilere hızlı ve basit bir şekilde web uygulamaları geliştirme imkanı sağlar. Flask, temel olarak bir çekirdek framework sunar ve ihtiyaç duyulan özellikler eklentilerle genişletilebilir. Bu esneklik, Flask'ı özellikle küçük projelerden büyük ve karmaşık uygulamalara kadar geniş bir yelpazede kullanılabilir hale getirir. Flask, hızlı prototipleme için idealdir ve Python'un güçlü kütüphane ekosistemini kullanarak kısa sürede işlevsel uygulamalar oluşturmayı kolaylaştırır. Jinja2 şablon motorunu kullanarak dinamik HTML sayfaları oluşturabilir, URL yönlendirmeleri ile farklı sayfalara veya API uç noktalarına erişim sağlayabilirsiniz. Ayrıca, WSGI (Web Server Gateway Interface) uyumluluğu sayesinde Flask uygulamaları, herhangi bir WSGI uyumlu sunucuda sorunsuz bir şekilde çalıştırılabilir. Bu özellikleriyle Flask, hem yeni başlayanlar hem de deneyimli geliştiriciler için güçlü bir web geliştirme aracı olarak öne çıkar.

C.1 Üretim ortamında Gunicorn ile çalıştırmak için:**bash****gunicorn-w 4 app:app**

Gunicorn (Green Unicorn), Python programlama dili ile yazılmış, WSGI (Web Server Gateway Interface) uyumlu bir HTTP sunucusudur. Flask ve Django gibi Python web framework'leri ile çalışan web uygulamalarını üretim ortamında daha güvenli ve verimli bir şekilde çalıştırmak için kullanılır. Basit, hafif ve hızlı bir yapıya sahip olan Gunicorn, özellikle yüksek performans gerektiren web uygulamaları için tercih edilir.

Gunicorn, WSGI uyumlu olduğu için Python tabanlı web uygulamalarını web sunucularına entegre eder ve çoklu iş parçacığı (multi-threading) desteği sayesinde aynı anda birden fazla isteği işleyebilir. Bu, uygulamanın performansını artırarak daha fazla kullanıcının eş zamanlı olarak erişim sağlamasını mümkün kılar. Ayrıca, ölçeklenebilirlik açısından esnektir ve kolayca yapılandırılabilir. Bu nedenle, geliştiriciler Gunicorn'u Flask ve Django gibi framework'lerle birlikte kullanarak web uygulamalarını üretim ortamında sorunsuz bir şekilde çalıştırabilirler.

Çalışma Tarihi: 01/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

Ancak windowsta gunicorn kullanılamadığı için waitress tercih ettim. Çünkü Gunicorn, Unix tabanlı sistemler için (Linux ve macOS gibi) tasarlanmış bir WSGI HTTP sunucusudur. Waitress de alternatif bir çözümdür.

Waitress: Flask veya Django gibi WSGI tabanlı Python web uygulamalarını Windows'ta çalıştırmak için uygun bir seçenektir. Waitress, Gunicorn gibi bir WSGI sunucusudur ve Windows'ta sorunsuz çalışır.

```
C:\Users\ymö\Desktop\project-root>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it
ent. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 113-425-121
```

C:\Users\ymö\Desktop\project-root>python app.py: Bu komut, app.py dosyasını Python ile çalıştırmanızı sağlar.

Serving Flask app 'app': Bu, Flask uygulamasının başarıyla başlatıldığını gösterir.

Debug mode: on: Bu, Flask uygulamasının debug (hata ayıklama) modunda çalıştığını gösterir. Debug modu, hataları hızlıca tespit edebilmeniz için uygulamayı daha ayrıntılı bir şekilde izlemenizi sağlar.

Running on <http://127.0.0.1:5000>: Flask uygulamanızın çalıştığı yerel IP adresi ve portu gösterir. 127.0.0.1, genellikle "localhost" olarak bilinir ve sadece bilgisayarınızdaki bağlantıları temsil eder. 5000 ise Flask'ın varsayılan portudur. Tarayıcınızı açarak <http://127.0.0.1:5000> adresine gidip uygulamayı görüntüleyebilirsiniz.

Press CTRL+C to quit: Flask uygulamasını durdurmak için klavyede CTRL+C tuşlarına basmanız gerektiğini belirtir.

Çalışma Tarihi: 02/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

Restarting with watchdog (windowsapi): Flask, debug modundayken, kodda bir değişiklik olduğunda uygulamanızı otomatik olarak yeniden başlatır. "Watchdog" adı verilen bu mekanizma, Python kod dosyalarını izler ve bir değişiklik tespit ettiğinde uygulamayı yeniden başlatır. (windowsapi) ifadesi, bu işlemin Windows ortamında Windows API'si kullanılarak yapıldığını gösterir.

Debugger is active!: Bu, Flask'ın hata ayıklama özelliğinin aktif olduğunu gösterir. Eğer uygulamada bir hata oluşursa, bu mod sayesinde ayrıntılı hata raporları ve hata izleme (traceback) bilgileri alabilirsiniz.

Debugger PIN: 113-425-121: Flask'ın debug arayüzüne tarayıcıdan erişmek için kullanabileceğiniz bir PIN numarasıdır. Bu, yetkisiz erişimleri engellemek için kullanılır. Ancak, bu özellik daha çok sunucu ortamlarında gereklidir ve genellikle yerel geliştirme sırasında kullanılmaz.

4. Flask API

A. app.py

Flask uygulamasının ana dosyası.

İçerik:

- from flask import Flask, request, jsonify: Flask ve gerekli modülleri içeri aktarın.
- app = Flask(__name__): Flask uygulamasını başlatın.
- from model import predict: Yapay zeka modelini içeri aktarın.

```
import logging
from flask import Flask, request, jsonify
from flask_jwt_extended import JWTManager, create_access_token, jwt_required
from logging.handlers import RotatingFileHandler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import joblib
import pandas as pd
from datetime import datetime, timedelta

app = Flask(__name__)
```

Çalışma Tarihi: 05/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

B. predict Endpoint

API isteğini alır, veriyi yapay zeka modeli ile işler ve sonucu geri döner.

İçerik:

```
@app.route('/predict', methods=['POST'])
```

```
def predict():
```

```
    data = request.get_json()
```

```
    account_id = data['accountId']
```

```
    transactions = data['transactions']
```

```
    predictions = predict(transactions)
```

```
    return jsonify({'predictions': predictions})
```

```
@app.route('/predict', methods=['POST'])
@jwt_required()
def predict_route():
    try:
        data = request.get_json()
        account_id = data['accountId']
        transactions = data['transactions']

        predictions = predict(transactions)

        return jsonify({'predictions': predictions})
    except KeyError as e:
        return jsonify(error=f"Gerekli veri eksik: {str(e)}"), 400
    except Exception as e:
        return jsonify(error=str(e)), 500
```

@app.route('/predict', methods=['POST']): Bu satır, /predict endpoint'inin POST isteklerini kabul edeceğini belirtir.

def predict_route(): Bu, predict fonksiyonunun başlangıç noktasıdır.

data = request.json: İstekten gelen JSON veriyi alır ve data değişkenine atar.

predictions = predict(data): Bu satır, data içinde bulunan veriyi kullanarak predict fonksiyonunu çağırır ve tahminleri döndürür.

return jsonify({'predictions': predictions}): Tahmin sonuçlarını JSON formatında döndürür.

Çalışma Tarihi: 06/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

5. Yapay Zeka Modeli

A. model.py

Yapay zeka modelini içeren dosya. Bu dosyada verilerin işlenmesi ve modelin öngörü yapması sağlanır.

İçerik:

- import pandas as pd: Pandaskütüphanesini içeri aktarın.
- def predict(transactions): Verileri işleyen ve öngörü sağlayan fonksiyon.

```
app.py* X model.py X
import pandas as pd

def predict(data):
    transactions = data.get('transactions', [])
    df = pd.DataFrame(transactions)

    if 'amount' not in df.columns:
        raise KeyError("DataFrame'de 'amount' sütunu bulunmuyor.")

    df['prediction'] = df['amount'].apply(lambda x: 'high' if x > 100 else 'low')
    return df.to_dict(orient='records')

if __name__ == "__main__":
    test_data = [
        {'transaction_amount': 250},
        {'transaction_amount': 50},
        {'transaction_amount': 400},
        {'transaction_amount': None}
    ]

    predictions = predict(test_data)
    print(predictions)
```

predict Fonksiyonu: predict fonksiyonu, data adlı bir parametre alır. Bu parametre, işlemleri içeren bir veri kümesi olarak kabul edilir. data içindeki transactions anahtarıyla ilişkili olan işlemler, bir liste olarak alınır ve bu liste bir Pandas DataFrame'e dönüştürülür. Eğer DataFrame'in amount adlı bir sütunu yoksa, fonksiyon bir KeyError fırlatır. Bu, işlemleri doğru şekilde işlemek için gerekli bir sütunun eksik olduğunu belirtir. DataFrame oluşturulduktan sonra, her bir işlemin miktarına (amount) bağlı olarak bir tahmin yapılır. Eğer miktar 100'den büyükse, tahmin high (yüksek) olarak, değilse low (düşük) olarak belirlenir. Bu tahminler yeni bir sütunda saklanır: prediction. Son olarak, tahmin edilen değerler içeren DataFrame, bir Python sözlüğü listesi olarak dönüştürülür ve fonksiyon bu listeyi döner.

Çalışma Tarihi: 07/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

__main__ Bloğu: Bu blok, fonksiyonun doğrudan bu dosya çalıştırıldığında nasıl kullanılacağını gösterir. test_data adında örnek bir veri kümesi oluşturulur. Bu veri kümesi, işlem miktarlarını içeren birkaç sözlük içerir.

predict fonksiyonu bu veri kümesi üzerinde çalıştırılır ve sonuç olarak her bir işlem için tahminler döner. Bu tahminler, print(predictions) ifadesiyle ekrana yazdırılır.

Özetle, bu kod, işlem miktarına dayalı olarak basit tahminler yapan ve bu tahminleri bir liste olarak döndüren bir işlemi gerçekleştirir.

Örnek Fonksiyon:

python

```
def predict(transactions):
```

```
    df = pd.DataFrame(transactions)
```

```
    # Örneğin, basit bir öngörü modeli ile dummy değerler döndürün
```

```
    df['prediction'] = df['amount'].apply(lambda x: 'high' if x > 100 else 'low')
```

```
    return df.to_dict(orient='records')
```

6. Örnek İstek ve Yanıtlar

A. Örnek İstek

json

```
{
  "accountId": 12345,
  "transactions": [
    { "timestamp": "2023-07-27T10:00:00", "amount": 150.0, "type": "debit" },
    { "timestamp": "2023-07-27T11:00:00", "amount": 20.0, "type": "credit" }
  ]
}
```

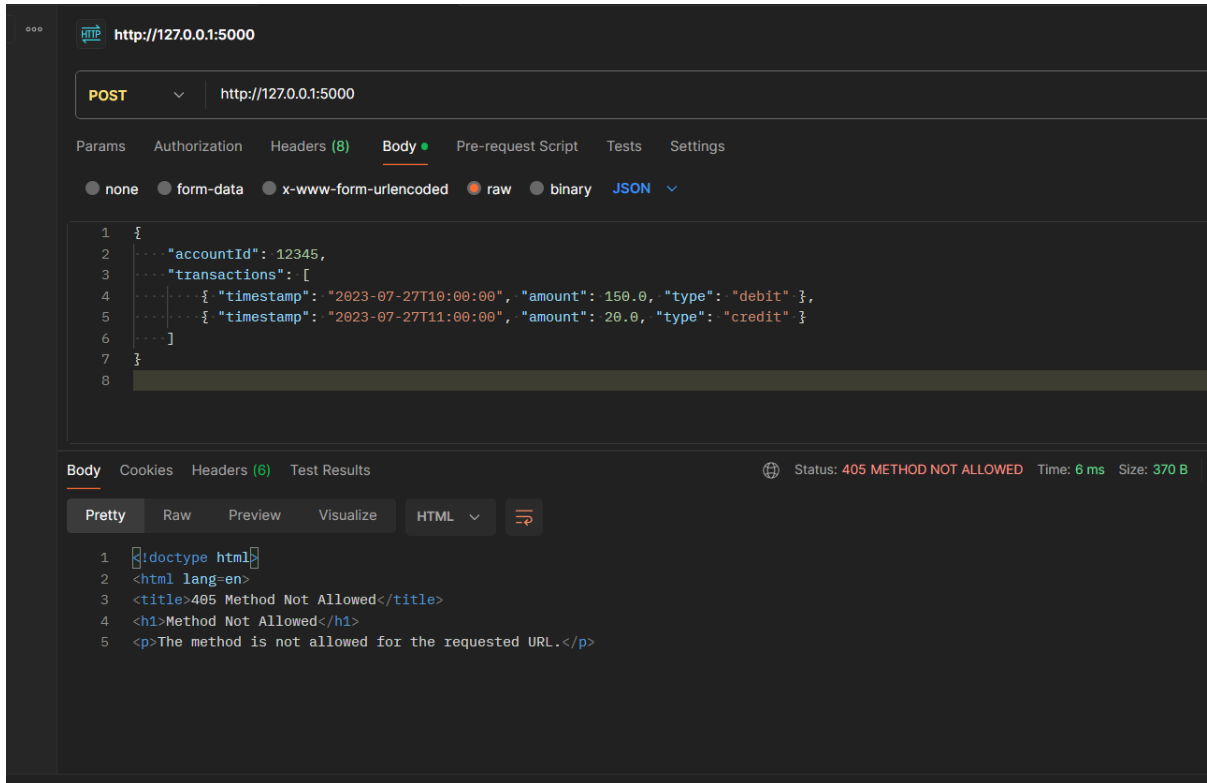
Çalışma Tarihi: 08/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

B. Örnek Yanıt

Json

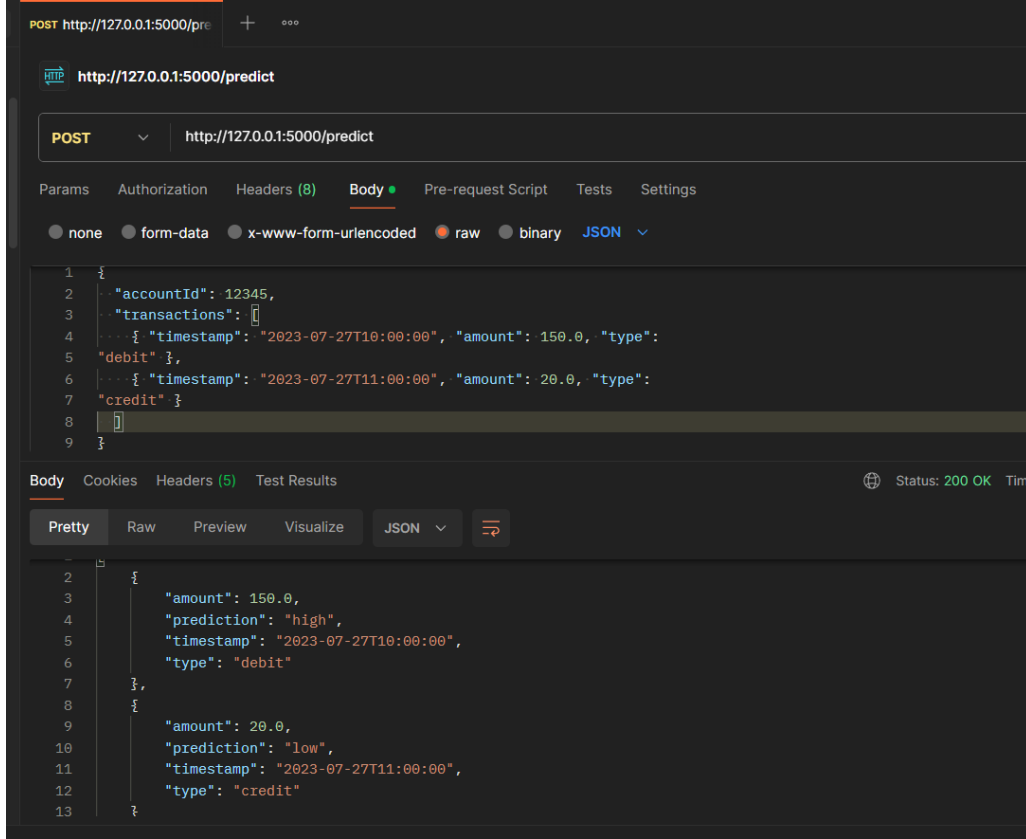
```
{
  "predictions": [
    { "timestamp": "2023-07-27T10:00:00", "amount": 150.0, "type": "debit", "prediction":
"high" },
    { "timestamp": "2023-07-27T11:00:00", "amount": 20.0, "type": "credit", "prediction":
"low" }
  ]
}
```



Bu hatayı aldım.

Çalışma Tarihi: 09/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....



Postman kullanılarak yerel bir Flask API'sine yapılan bir POST isteği.

İstek Detayları:

URL: <http://127.0.0.1:5000/predict>

Bu, Flask API'sinde predict adlı bir rota olduğunu ve bu rotaya bir POST isteği gönderildiğini gösterir.

İstek Yöntemi: POST

İstek, JSON formatında veri göndererek yapılıyor.

Çalışma Tarihi: 12/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

Body Bölümü: JSON formatında gönderilen veri şu şekilde:

```
{  
  "accountId": 12345,  
  "transactions": [  
    { "timestamp": "2023-07-27T10:00:00", "amount": 150.0, "type": "debit" },  
    { "timestamp": "2023-07-27T11:00:00", "amount": 20.0, "type": "credit" }  
  ]  
}
```

accountId: 12345 - Bu, hesaba ait bir kimlik numarasını temsil eder.

transactions: Bir işlem listesi içerir. Her işlem, bir zaman damgası, miktar (amount), ve işlem türünü (type) içerir.

- İlk işlem, 150.0 birim miktarında bir "debit" (borç) işlemidir ve 2023-07-27 tarihinde saat 10:00'da gerçekleşmiştir.
- İkinci işlem, 20.0 birim miktarında bir "credit" (alacak) işlemidir ve 2023-07-27 tarihinde saat 11:00'da gerçekleşmiştir.

Yanıt Detayları:

Yanıt Durumu: 200 OK Bu, isteğin başarıyla işlendiğini gösterir. **Yanıt Body'si:** JSON formatında şu şekilde döndü:

```
{  
  "amount": 150.0,  
  "prediction": "high",  
  "timestamp": "2023-07-27T10:00:00",  
  "type": "debit"  
  
  "amount": 20.0,  
  "prediction": "low",  
  "timestamp": "2023-07-27T11:00:00",  
  "type": "credit"  
}
```

Çalışma Tarihi: 13/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

7. Hata Yönetimi ve Logging

A. Hata Yönetimi

- Flask ile global hata yönetimi sağlanabilir.
- Örnek: `@app.errorhandler(Exception)` kullanarak genel bir hata yakalayıcı ekleyin.

B. Logging

- Flask'ın dahili logging mekanizmasını kullanabilirsiniz.
- Örnek: `app.logger.error('An error occurred')`

```
@app.errorhandler(Exception)
def handle_exception(e):
    app.logger.error(f"Bir hata oluştu: {e}")
    return jsonify(error=str(e)), 500

@app.route('/login', methods=['POST'])
def login():
    username = request.json.get('username', None)
    password = request.json.get('password', None)

    if username != 'admin' or password != 'password':
        return jsonify({"msg": "Kullanıcı adı veya şifre hatalı"}), 401

    access_token = create_access_token(identity=username)
    return jsonify(access_token=access_token)
```

1. Hata Yönetimi (Error Handling)

@app.errorhandler(Exception): Bu dekoratör, Flask uygulamasında oluşabilecek genel hataları yakalamak için kullanılır.

def handle_exception(e): Bu fonksiyon, yakalanan hatayı işler.

app.logger.error(f"Bir hata oluştu: {e}"): Hata mesajı, uygulama logger'ına kaydedilir.

return jsonify(error=str(e)), 500: Hata mesajı, JSON formatında döndürülür ve HTTP 500 (Internal Server Error) kodu ile yanıt verilir.

Etiya stajyerlerle cv nasıl hazırlanır hakkında bilgilendirme toplantısı yapıldı.

Çalışma Tarihi:
14/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

2. Login Rotası (Login Route)

@app.route('/login', methods=['POST']): Bu dekoratör, /login yoluna bir POST isteği yapılmasını sağlar. Bu, kullanıcıların oturum açması için kullanılan bir API rotasıdır.

def login(): Bu fonksiyon, kullanıcı kimlik doğrulama işlemini gerçekleştirir.

username = request.json.get('username', None): İstek gövdesinden (request.json) "username" değeri alınır. Eğer yoksa None değeri döner.

password = request.json.get('password', None): İstek gövdesinden "password" değeri alınır. Eğer yoksa None değeri döner.

if username != 'admin' or password != 'password': Eğer kullanıcı adı "admin" değilse veya şifre "password" değilse, oturum açma başarısız sayılır.

return jsonify({"msg": "Kullanıcı adı veya şifre hatalı"}), 401: Yanıt olarak bir hata mesajı döndürülür ve HTTP 401 (Unauthorized) kodu ile yanıt verilir.

access_token = create_access_token(identity=username): Eğer kullanıcı adı ve şifre doğruysa, kullanıcı kimliği (username) ile bir access_token (erişim belirteci) oluşturulur.

return jsonify(access_token=access_token): Oluşturulan erişim belirteci, JSON formatında döndürülür.

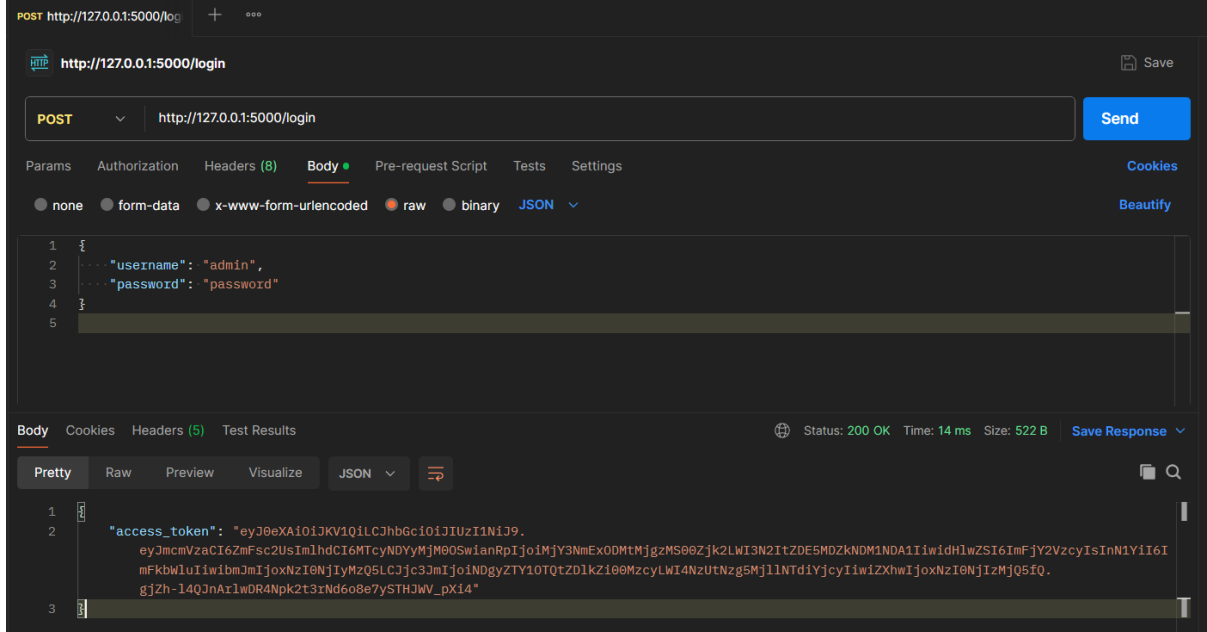
Genel olarak yorumlayacak olursak, bir kullanıcının geçerli kimlik bilgileriyle oturum açmasını sağlayan basit bir login mekanizması içerir. Eğer kullanıcı adı veya şifre yanlışsa, buna uygun bir hata mesajı döndürülür. Ayrıca, herhangi bir beklenmeyen hata oluştuğunda, bu hatayı yakalayarak detaylı bir hata mesajı ile yanıt verir. Bu mekanizma, uygulamanın daha güvenli ve hata toleranslı olmasını sağlar.

Çalışma Tarihi: 15/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

8. Güvenlik

- JWT veya API anahtarları kullanarak API erişimini güvence altına alın.
- Flask-JWT-Extended veya Flask-HTTPAuth gibi kütüphaneler kullanılabilir.



İstek /login rotasına gönderilmiştir ve JSON formatında kullanıcı adı (username) ve şifre (password) bilgilerini içermektedir.

1. POST İsteği:

URL: <http://127.0.0.1:5000/login> Bu, yerel sunucuda çalışan bir Flask uygulamasının /login rotasına yapılan bir POST isteğidir

İstek Gövdesi (Body):

Raw JSON formatında

```
{
  "username": "admin",
  "password": "password"
}
```

Bu gövde, kullanıcının kimlik bilgilerini içerir. Bu bilgiler, API tarafından doğrulanacak

Çalışma Tarihi: 16/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

2. İstek Yanıtı:

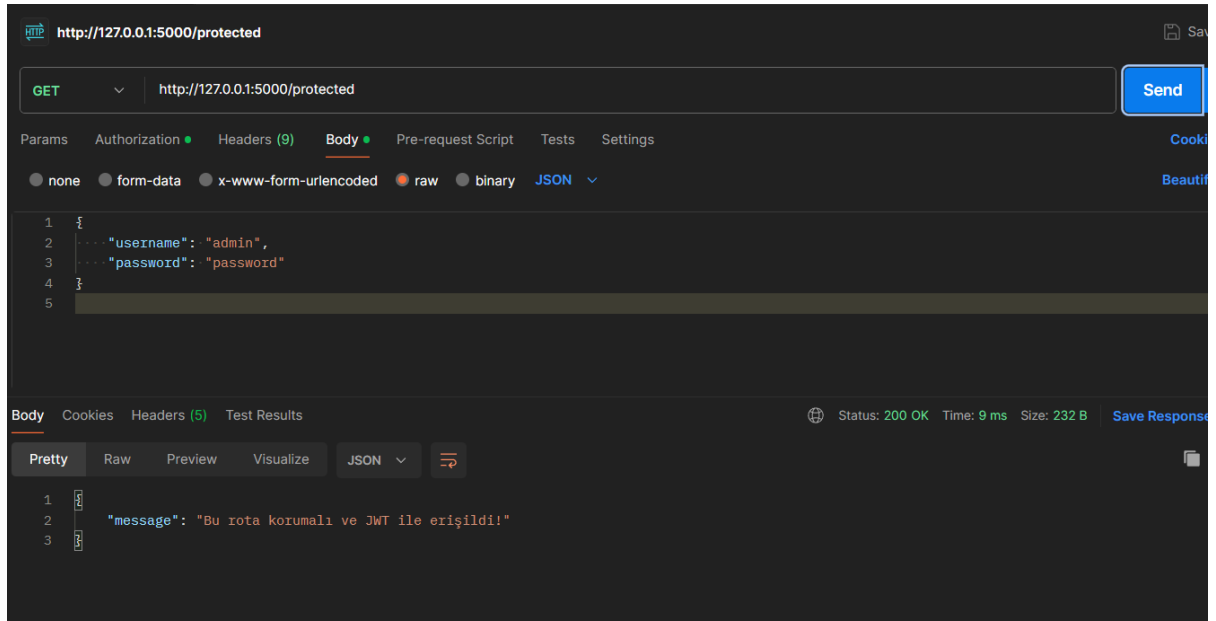
Yanıt Durumu: 200 OK ; Bu durum kodu, isteğin başarıyla işlendiğini gösterir.

Yanıt Gövdesi (Body): Yanıt olarak, JSON formatında bir access_token döndürülmüştür:

```
{  
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."  
}
```

access_token: Bu, kullanıcıya verilen ve API'ye yapılacak gelecekteki taleplerde kullanılacak olan bir JSON Web Token (JWT) olarak bilinir. Bu token, kullanıcının kimliğini doğrulamak ve yetkilendirmek için kullanılır.

Bu işlem, bir kullanıcının başarılı bir şekilde oturum açmasını ve bir JWT token almasını sağlamıştır. Bu token, API'deki diğer korunan rotalara erişmek için kullanılabilir. Örneğin, kullanıcı bu token'ı, kimlik doğrulaması gerektiren başka bir API isteğinde Authorization başlığı altında iletebilir. Bu token, genellikle belirli bir süre boyunca geçerlidir ve bu süre sona erdiğinde yenilenmesi gerekir.



Çalışma Tarihi: 19/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

Etiya Bilgi Teknolojileri Yazılım San ve Tic A.Ş	Sayfa No: 22
<p>İstek Detayları:</p> <p>İstek türü GET ve /protected rotasına yönlendirilmiştir.</p> <p>Bu istek, http://127.0.0.1:5000/protected URL'sine yapılmıştır.</p> <p>İstek gövdesinde (Body), username ve password içeren bir JSON verisi gönderilmiştir.</p> <p>Yanıt Detayları:</p> <p>Yanıtın HTTP durumu 200 OK, yani istek başarıyla işlenmiştir.</p> <p>Yanıta JSON formatında bir mesaj döndürülmüş: "Bu rota korumalı ve JWT ile erişildi!".</p> <p>Bu durum, korumalı bir rotaya erişmek için bir JWT (JSON Web Token) doğrulaması yapıldığını ve doğru kimlik bilgileri sağlanarak erişimin başarıyla gerçekleştirildiğini gösterir. JWT doğrulaması geçtikten sonra rota çalıştırılmış ve ilgili mesaj döndürülmüştür.</p> <p>Yapay Zeka Modeli</p> <p>Yapay Zeka Modeli Geliştirme ve Kullanma Adımları</p> <ol style="list-style-type: none">1. Veri Toplama ve Hazırlama <p>Modelinizi eğitmek için gerekli verileri toplayın ve hazırlayın. Bu veriler genellikle hesap hareketleri ve ilgili diğer finansal bilgiler olacaktır.</p> <ol style="list-style-type: none">2. Veri Ön İşleme <p>Verileri analiz edilebilir hale getirmek için temizleyin ve dönüştürün. Örneğin:</p> <ul style="list-style-type: none">• Eksik değerleri doldurun veya çıkarın.• Tarih ve zaman bilgilerini dönüştürün.• Kategorik verileri sayısal verilere dönüştürün (One-hot encoding gibi). <ol style="list-style-type: none">3. Model Geliştirme <p>Yapay zeka modelini oluşturun ve eğitin. Bu örnekte, Scikit-learn kullanarak basit bir model eğiteceğiz.</p> <ol style="list-style-type: none">4. Modeli Kaydetme ve Yükleme <p>Eğitilmiş modeli disk üzerinde kaydedin ve Flask API'de kullanmak üzere yükleyin.</p> <ol style="list-style-type: none">5. Modeli Kullanarak Öngörü Yapma <p>API üzerinden gelen verileri modele verip öngörüler alın ve kullanıcıya geri döndürün</p>	
Çalışma Tarihi: 20/08/2024	Onaylayanın, Adı – Soyadı: Elif AKGÜN İmzası:.....

Örnek Uygulama Süreci

A. Veri Toplama ve Hazırlama

Veri setinizi transactions.csv isimli bir CSV dosyasına kaydedin. Bu dosya hesap hareketlerini içerecektir.

B. Veri Ön İşleme

python

```
import pandas as pd
```

```
# Veriyi yükleyin
```

```
df = pd.read_csv('transactions.csv')
```

```
# Tarih ve zaman bilgilerini dönüştürün
```

```
df['timestamp'] = pd.to_datetime(df['timestamp'])
```

```
# Kategorik verileri sayısal verilere dönüştürün
```

```
df = pd.get_dummies(df, columns=['type'])
```

```
app.py × model.py ×
import pandas as pd

def predict(data):
    transactions = data.get('transactions', [])
    df = pd.DataFrame(transactions)

    if 'amount' not in df.columns:
        raise KeyError("DataFrame'de 'amount' sütunu bulunmuyor.")

    df['prediction'] = df['amount'].apply(lambda x: 'high' if x > 100 else 'low')
    return df.to_dict(orient='records')

if __name__ == "__main__":
    test_data = [
        {'transaction_amount': 250},
        {'transaction_amount': 50},
        {'transaction_amount': 400},
        {'transaction_amount': None}
    ]

    predictions = predict(test_data)
    print(predictions)
```

Çalışma Tarihi: 21/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

C. Model Geliştirme

python

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import joblib ,
# Özellikler ve hedef değişkeni belirleyin
X = df.drop(['prediction'], axis=1)
y = df['prediction']
# Eğitim ve test setlerine ayırın
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Modeli eğitin
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
# Modeli kaydedin
joblib.dump(model, 'model.pkl')
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import joblib
```

```
def train_model(df):
    if 'prediction' in df.columns:
        X = df.drop(['prediction'], axis=1)
        y = df['prediction']
    else:
        raise KeyError("DataFrame'de 'prediction' sütunu bulunamadı.")

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    joblib.dump(model, 'model.pkl')
    return model
```

Çalışma Tarihi: 22/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

D. Flask API'de Modeli Kullanma

app.py Dosyası:

python

from flask import Flask, request, jsonify

import joblib

import pandas as pd

app = Flask(__name__)

Modeli yükleyin

model = joblib.load('model.pkl')

def preprocess_data(transactions):

Verileri DataFrame'e dönüştürün

df = pd.DataFrame(transactions)

Gerekli veri ön işlemlerini yapın

df['timestamp'] = pd.to_datetime(df['timestamp'])

df = pd.get_dummies(df, columns=['type'])

return df

@app.route('/predict', methods=['POST'])

def predict():

data = request.get_json() transactions = data['transactions']

df = preprocess_data(transactions)

predictions = model.predict(df)

df['prediction'] = predictions

return jsonify(df.to_dict(orient='records'))

if __name__ == '__main__':

app.run(debug=True)

Çalışma Tarihi: 23/08/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

```

@app.route('/protected', methods=['GET'])
@jwt_required()
def protected():
    return jsonify(message="Bu rota korumalı ve JWT ile erişildi!")

@app.route('/predict', methods=['POST'])
@jwt_required()
def predict_route():
    try:
        data = request.get_json()
        account_id = data['accountId']
        transactions = data['transactions']

        predictions = predict(transactions)

        return jsonify({'predictions': predictions})
    except KeyError as e:
        return jsonify(error=f"Gerekli veri eksik: {str(e)}"), 400
    except Exception as e:
        return jsonify(error=str(e)), 500

def generate_data(num_entries):
    timestamps = [datetime.now() - timedelta(minutes=30*i) for i in range(num_entries)]
    amounts = [round(100 * i, 2) for i in range(num_entries)]
    types = ['debit' if i % 2 == 0 else 'credit' for i in range(num_entries)]

    data = {
        'timestamp': [ts.isoformat() for ts in timestamps],
        'amount': amounts,
        'type': types
    }

    df = pd.DataFrame(data)
    df.to_csv('transactions.csv', index=False)
    print("transactions.csv dosyası başarıyla oluşturuldu.")

```

```

def preprocess_data(filepath):
    df = pd.read_csv(filepath)
    print("Original veri türleri:")
    print(df.dtypes)

    if 'timestamp' in df.columns:
        df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')
        df['timestamp'] = df['timestamp'].astype('int64') // 10**9

    if 'type' in df.columns:
        df = pd.get_dummies(df, columns=['type'])

    bool_cols = df.select_dtypes(include=['bool']).columns
    if not bool_cols.empty:
        df[bool_cols] = df[bool_cols].astype(int)

    if df.isnull().sum().any():
        print("Eksik veriler bulundu. Eksik verileri dolduruyoruz.")
        df = df.fillna(0)

    print("Dönüştürülmüş veri türleri:")
    print(df.dtypes)

    return df

def train_model(df):
    if 'prediction' in df.columns:
        X = df.drop(['prediction'], axis=1)
        y = df['prediction']
    else:
        raise KeyError("DataFrame'de 'prediction' sütunu bulunamadı.")

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    joblib.dump(model, 'model.pkl')

```

Çalışma Tarihi: 02/09/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

```
def predict(data):

    model = joblib.load('model.pkl')

    df = pd.DataFrame(data['transactions'])

    df['timestamp'] = pd.to_datetime(df['timestamp']).astype('int64') // 10**9
    df = pd.get_dummies(df, columns=['type'])

    if 'type_credit' not in df.columns:
        df['type_credit'] = 0
    if 'type_debit' not in df.columns:
        df['type_debit'] = 0

    predictions = model.predict(df)

    df['prediction'] = predictions

    response = []
    for i, row in df.iterrows():
        result = {
            "timestamp": datetime.fromtimestamp(row['timestamp']).isoformat(),
            "amount": row['amount'],
            "type": "debit" if row['type_debit'] else "credit",
            "prediction": "high" if row['prediction'] == 1 else "Low"
        }
        response.append(result)

    return response

@app.route('/train', methods=['POST'])
@jwt_required()
def train():
    try:

        generate_data(100)
```

```
generate_data(100)

df = preprocess_data(r'C:\Users\ymö\Desktop\project-root\transactions.csv')

df['prediction'] = df['amount'].apply(lambda x: 1 if x > 1000 else 0)

model = train_model(df)

return jsonify({"message": "Model başarıyla eğitildi ve kaydedildi."})
except Exception as e:
    return jsonify(error=str(e)), 500

if __name__ == "__main__":
    app.run(debug=True)
```

İstenilen komutlar ve bunların kodları.

Model Eğitimi Rotası (/train)

generate_data() ile sahte veriler oluşturulur.

Veriler preprocess_data() fonksiyonu ile işlenir ve sütunlar dönüştürülür.

train_model() fonksiyonu modeli eğitir ve kaydeder

Çalışma Tarihi: 03/09/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

Veri Üretme (generate_data)

- İşlem verilerini zaman damgası, tutar ve tür bilgileri ile üretir.

Veri Ön İşleme (preprocess_data)

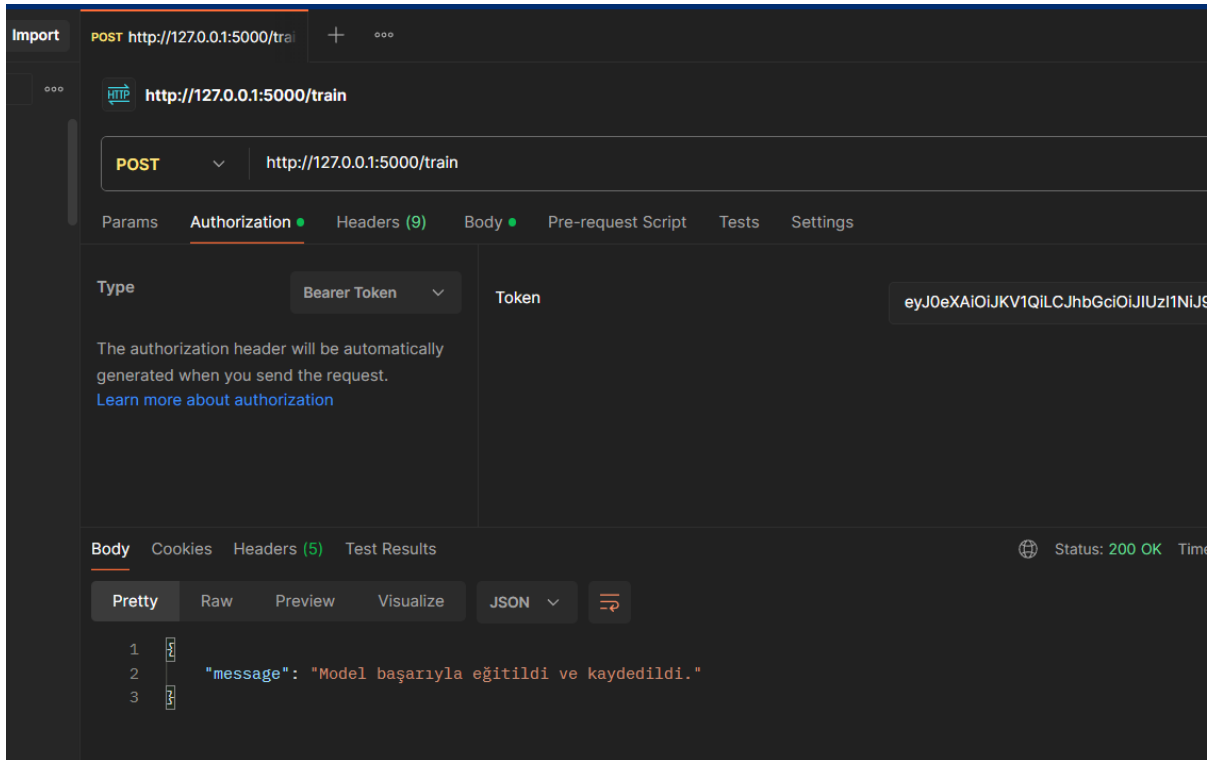
- Verileri temizler, dönüştürür ve analiz edilebilir hale getirir.

Model Eğitimi (train_model)

- RandomForestClassifier modelini eğitir ve model.pkl dosyasına kaydeder.

Tahmin Yapma (predict)

- Kaydedilmiş modeli yükler, işlem verileri üzerinde tahmin yapar ve sonucu JSON formatında döndürür.



Model eğitimini tamamladım.

Çalışma Tarihi: 04/09/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....

A	B	C	D	E
timestamp,amount,type				
2024-08-27T03:40:46.228278,0,debit				
2024-08-27T03:10:46.228278,100,credit				
2024-08-27T02:40:46.228278,200,debit				
2024-08-27T02:10:46.228278,300,credit				
2024-08-27T01:40:46.228278,400,debit				
2024-08-27T01:10:46.228278,500,credit				
2024-08-27T00:40:46.228278,600,debit				
2024-08-27T00:10:46.228278,700,credit				
2024-08-26T23:40:46.228278,800,debit				
2024-08-26T23:10:46.228278,900,credit				
2024-08-26T22:40:46.228278,1000,debit				
2024-08-26T22:10:46.228278,1100,credit				
2024-08-26T21:40:46.228278,1200,debit				
2024-08-26T21:10:46.228278,1300,credit				
2024-08-26T20:40:46.228278,1400,debit				
2024-08-26T20:10:46.228278,1500,credit				
2024-08-26T19:40:46.228278,1600,debit				
2024-08-26T19:10:46.228278,1700,credit				
2024-08-26T18:40:46.228278,1800,debit				
2024-08-26T18:10:46.228278,1900,credit				
2024-08-26T17:40:46.228278,2000,debit				
2024-08-26T17:10:46.228278,2100,credit				
2024-08-26T16:40:46.228278,2200,debit				
2024-08-26T16:10:46.228278,2300,credit				
2024-08-26T15:40:46.228278,2400,debit				
2024-08-26T15:10:46.228278,2500,credit				
2024-08-26T14:40:46.228278,2600,debit				
2024-08-26T14:10:46.228278,2700,credit				

Tamamlanan Transactions.csv dosyası içeriği. Model eğitildikten sonra;

6. Örnek İstek ve Yanıtlar

A. Örnek İstek

json

"accountId": 12345,

"transactions": [

{ "timestamp": "2023-07-27T10:00:00", "amount": 150.0, "type": "debit" },

{ "timestamp": "2023-07-27T11:00:00", "amount": 20.0, "type": "credit" }] }

B. Örnek Yanıt

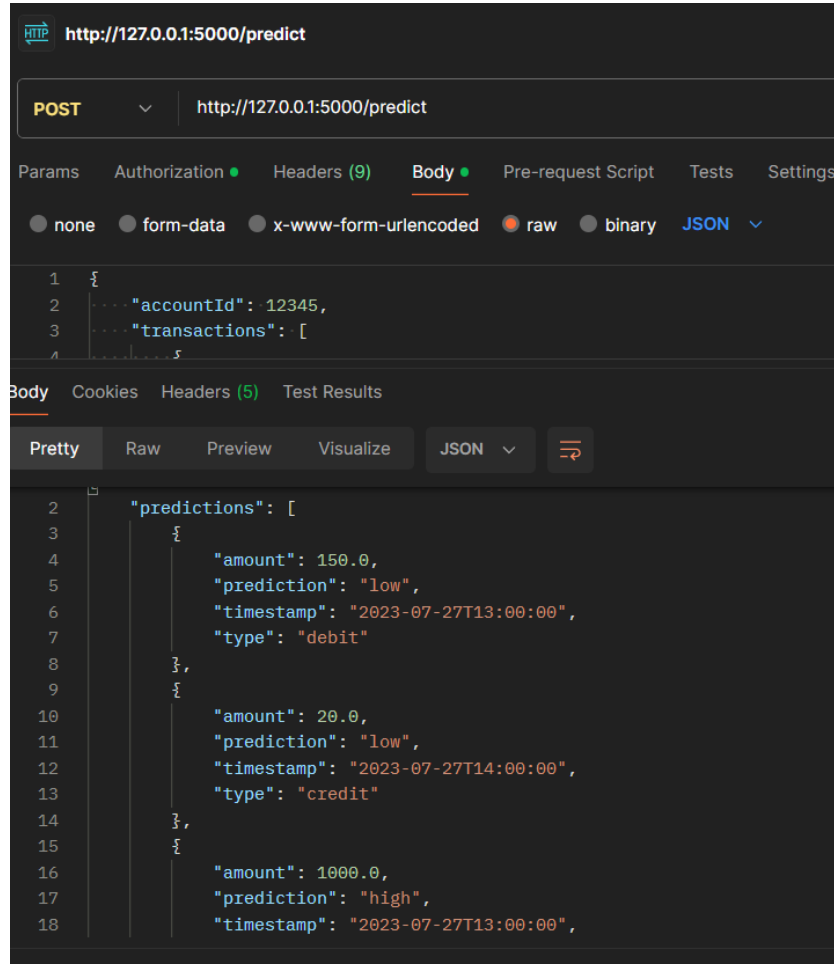
json { "predictions": [

{ "timestamp": "2023-07-27T10:00:00", "amount": 150.0, "type": "debit", "prediction": "high" },

{ "timestamp": "2023-07-27T11:00:00", "amount": 20.0, "type": "credit", "prediction": "low" }] }

Çalışma Tarihi: 05/09/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....



Model eğitimi tamamlandıktan sonra örnek istek postman üzerinde tekrar POST ve predict olarak URL gönderildi ve istenilen yanıt alındı. Proje istenildiği gibi tamamlandı ve staj rehberime sunum yaptım onayladı ve stajımı başarıyla tamamladım.

Çalışma Tarihi: 06/09/2024

Onaylayanın,
Adı – Soyadı: Elif AKGÜN
İmzası:.....