

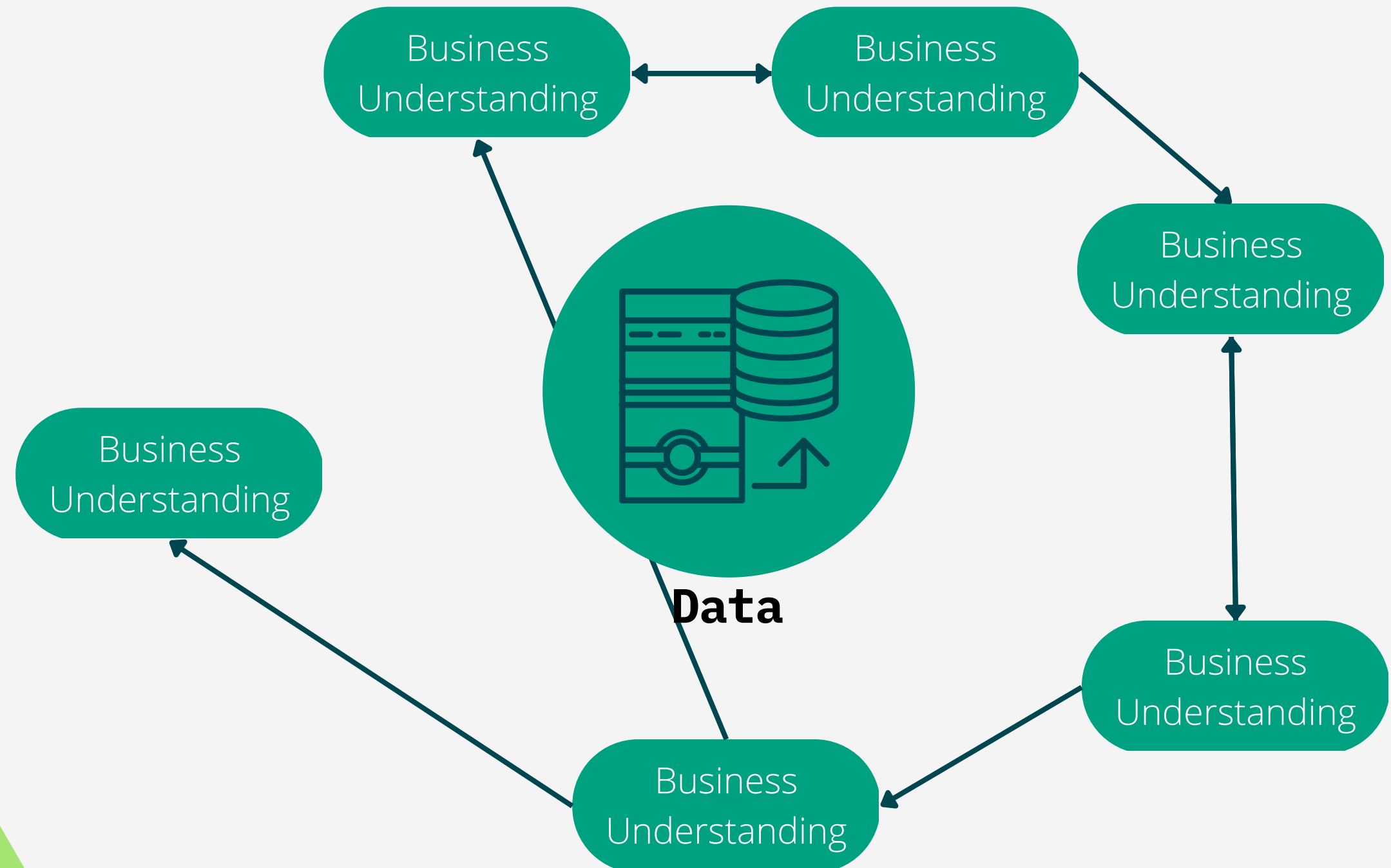
Uçtan Uca Makine Öğrenmesi Projesi



İçerik

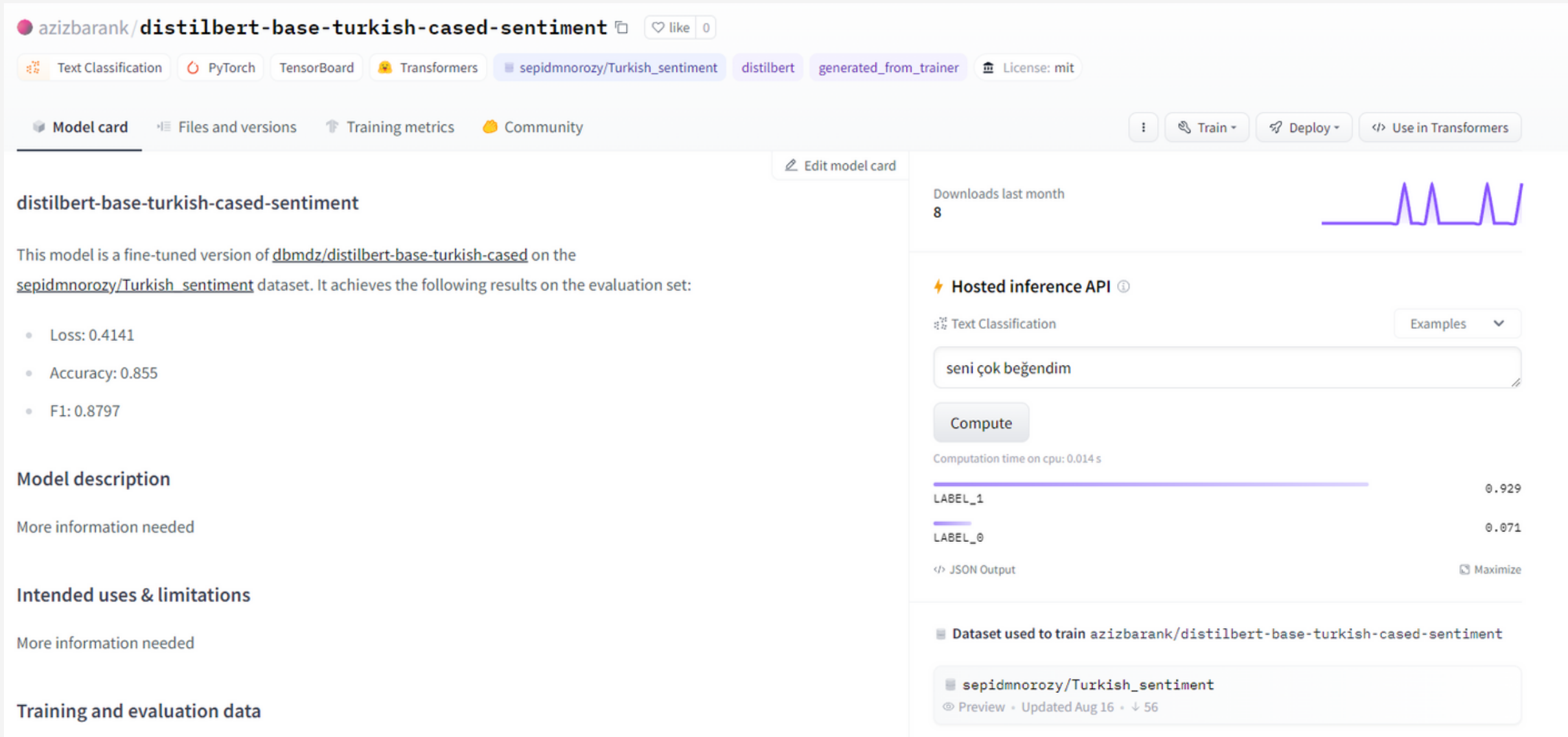
- CRISP-DM Diyagramı
 - Açıklamalar
- Duygu Analizi Örneği
 - HuggingFace
- FastAPI Nedir?
 - FastAPI vs Flask
 - Duygu analizi uygulamasının servis haline getirilmesi
 - Dockerization işlemi
- AWS
 - Hızlıca AWS servisleri
 - AWS'de canlıya çıkma

CRISP-DM



Duygu Analizi Örneđi

Örnek kapsamında hızlı olması açısından **HuggingFace** üzerinde herkese açık yayınlanmış bir duygu analizi modelini kullanacağız.



Modeli Kullanmak

Herkes açık modeli kullanmak için **transformers** kütüphanesinden yararlanıyoruz.

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer, pipeline

model = AutoModelForSequenceClassification.from_pretrained('azizbarank/distilbert-base-turkish-cased-sentiment')
tokenizer = AutoTokenizer.from_pretrained('azizbarank/distilbert-base-turkish-cased-sentiment')
pipeline = pipeline("sentiment-analysis", tokenizer=tokenizer, model=model)
pipeline('bu film çok iyiydi')
# {'label': 'LABEL_1', 'score': 0.8971970677375793}
```

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer, pipeline

class Sentiment_Analyser:
    def __init__(self):
        model = AutoModelForSequenceClassification.from_pretrained('azizbarank/distilbert-base-turkish-cased-sentiment')
        tokenizer = AutoTokenizer.from_pretrained('azizbarank/distilbert-base-turkish-cased-sentiment')
        self.pipeline = pipeline("sentiment-analysis", tokenizer=tokenizer, model=model)
        self.text_labels = {'LABEL_1': 'positive', 'LABEL_0': 'negative'}

    def predict(self, text):
        res = self.pipeline(text)[0]
        res['label'] = self.text_labels[res['label']]
        return res
```





Hızlı



Kolay



Otomatik

FastAPI Hız

Latency of 20-update responses, undefined

Framework	Average latency (lower is better)		σ (SD)	Max	Errors
■ blacksheep	403.9 ms	<div><div></div></div> 11.9%	97.6 ms	1400.0 ms	0
■ uvicorn	405.8 ms	<div><div></div></div> 12.0%	122.3 ms	1570.0 ms	0
■ starlette	409.7 ms	<div><div></div></div> 12.1%	241.4 ms	1930.0 ms	0
■ fastapi	409.9 ms	<div><div></div></div> 12.1%	173.6 ms	1610.0 ms	0
■ sanic	413.7 ms	<div><div></div></div> 12.2%	233.8 ms	2190.0 ms	0
■ responder	421.5 ms	<div><div></div></div> 12.5%	156.0 ms	1320.0 ms	0
■ tornado-py3-uvloop	441.1 ms	<div><div></div></div> 13.1%	186.8 ms	1630.0 ms	0
■ aiohttp-pg-raw	534.6 ms	<div><div></div></div> 15.8%	147.3 ms	1630.0 ms	0
■ api_hour-mysql	613.1 ms	<div><div></div></div> 18.1%	403.9 ms	2600.0 ms	10,705
■ api_hour	941.4 ms	<div><div></div></div> 27.9%	255.8 ms	1960.0 ms	0
■ flask-pypy2-raw	1340.0 ms	<div><div></div></div> 39.6%	277.8 ms	4270.0 ms	0
■ bottle-raw	1410.0 ms	<div><div></div></div> 41.7%	519.0 ms	4020.0 ms	0
■ tornado-pypy2	1420.0 ms	<div><div></div></div> 42.0%	72.3 ms	1810.0 ms	0
■ flask-raw	1460.0 ms	<div><div></div></div> 43.2%	536.8 ms	4050.0 ms	0

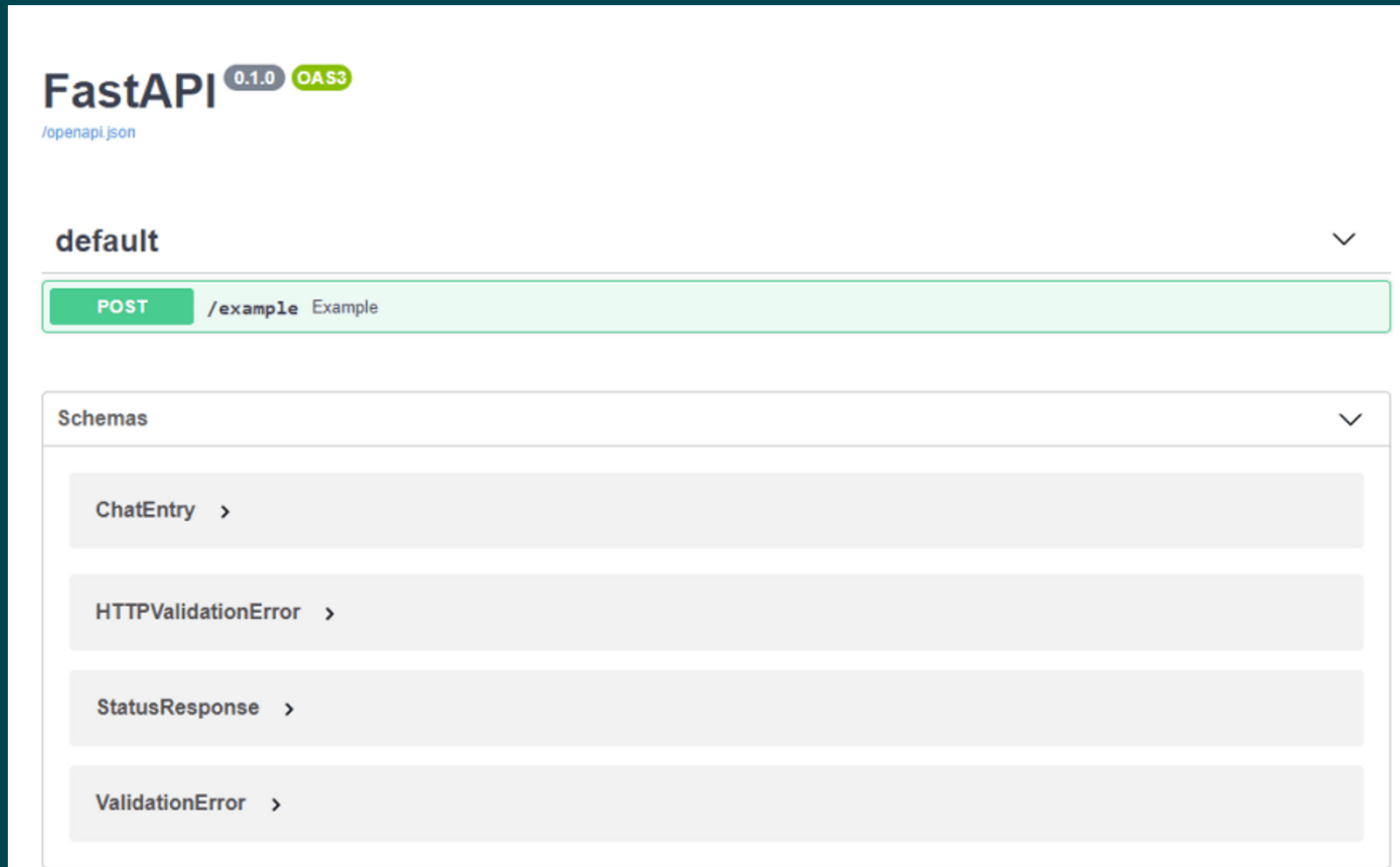
FastAPI Kolay

```
@app.route("/example/<uid>", methods=["POST"])
def handle_example(uid):
    arg = request.args.get('bleh')
    data = request.get_json()

    return jsonify(arg=arg, data=data, uid=uid)
```

```
@app.post("/example/{uid}")
def handle_example(uid: str, arg: str, data: MyData):
    return {"arg": arg, "data": data, "uid": uid}
```


FastAPI Oto



FastAPI vs Flask

FastAPI

- Asyncio ile varsayılan olarak destekli geliyor.
- Uygulamadaki servislerin birbirleri arasında konuşmasını sağlayan websocket desteğini varsayılan olarak sağlıyor.
- **Flask'a göre %300 daha hızlı**
- Otomatik Dokümantasyon oluşturmaları ve API'nin şemasını json çıktısı olarak verebilmesi

Flask API

- Asyncio desteği yok ancak Tornado ile asenkron istek alma olayını çözebiliyorsunuz
- WebSocket desteği için Tornado ile kullanmanız gerekiyor.
- Oldukça minimal Django gibi yapıların getirdiği karmaşıklığı ortadan kaldırıyor. Gereken yapıları siz tek tek yazıp ekliyorsunuz

Temel FastAPI

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def root():
    return {"message": "Hello World"}
```

```
$ uvicorn main:app --reload
```

```
INFO:      Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:      Started reloader process [28720]
INFO:      Started server process [28722]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
```

restart ↺

Duygu Analizi Servisi

Dosya Yapısı:

- `main.py` : Ana servislerin olduğu script
- `models.py` : İstek ve sonuçların döndüğü veri yapısı
- `transformer_model.py` : Duygu analizi fonksiyonu
- `requirements.txt` : Gerekli kütüphanelerin listelendiği dosya
- `Dockerfile` : Docker Container konfigürasyonu

models.py

```
from pydantic import BaseModel
from typing import List, Dict, Optional

class Argument(BaseModel):
    body: str
    domain: str

class ArgumentResponse(BaseModel):
    body: str
    evaluation: Dict
```

transformer_model.py

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer, pipeline

class Sentiment_Analyser:
    def __init__(self):
        model = AutoModelForSequenceClassification.from_pretrained('azizbarank/distilbert-base-turkish-cased-sentiment')
        tokenizer = AutoTokenizer.from_pretrained('azizbarank/distilbert-base-turkish-cased-sentiment')
        self.pipeline = pipeline("sentiment-analysis", tokenizer=tokenizer, model=model)
        self.text_labels = {'LABEL_1': 'positive', 'LABEL_0': 'negative'}

    def predict(self, text):
        res = self.pipeline(text)[0]
        res['label'] = self.text_labels[res['label']]
        return res
```

main.py

```
from fastapi import FastAPI, Body, HTTPException, status, Depends
from starlette.requests import Request

from models import Argument,\
    ArgumentResponse

from transformer_model import Sentiment_Analyser

CONSTANTS = {
    "statusResponse": {
        "status": "Sentiment API up and running!"
    },
    "sentimentExample": {
        "body": "seni çok seviyorum",
        "domain": "general"
    }
}

tags_metadata = [
    {
        "name": "Status",
    },
    {
        "name": "Evaluations",
        "description": "Operations to Make Predictions for text.",
    }
]
```


main.py

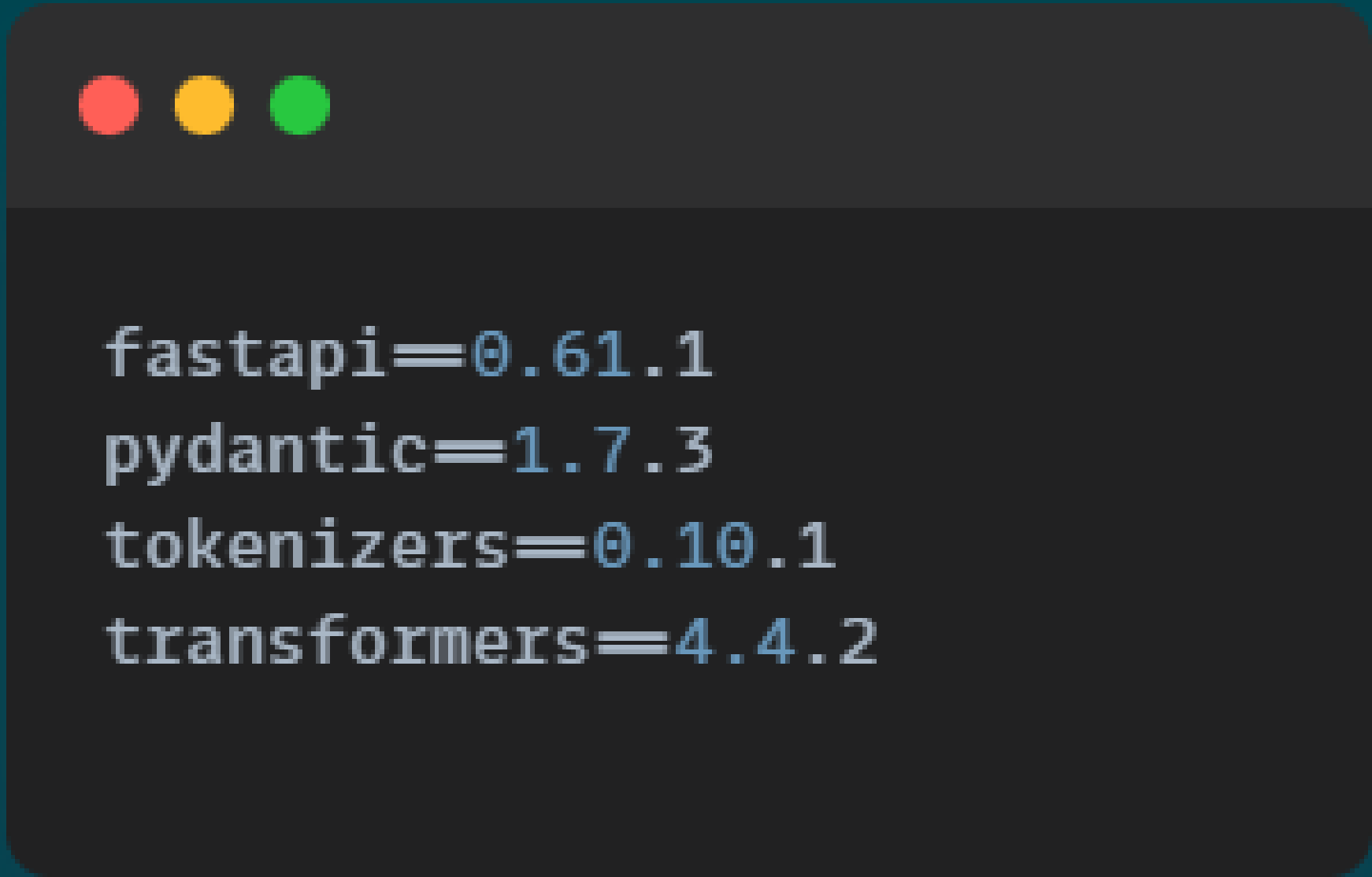
```
app = FastAPI(
    title= "Sentiment App",
    version="0.1",
    description="Simple Service for NLP",
    openapi_tags=tags_metadata
)

global sentiment_model
sentiment_model = SentimentAnalyser()

@app.get('/status', tags=["Status"])
def status_check():
    return CONSTANTS["statusResponse"]

@app.post('/sentiment-analysis', summary = "Sentiment Analysis for Text", status_code = 200, tags=["Evaluations"])
def sentiment_eva(request: Request, argument: Argument = Body(..., example = CONSTANTS["sentimentExample"])):
    evaluation = sentiment_model.predict(argument.body)
    response = ArgumentResponse(body = argument.body, evaluation = evaluation)
    return response
```

requirements.txt



```
fastapi=0.61.1  
pydantic=1.7.3  
tokenizers=0.10.1  
transformers=4.4.2
```

Sonuç

Sentiment App 0.1 OAS3

/openapi.json

Simple Service for NLP

Status

GET

/status Status Check

Evaluations Operations to Make Predictions for text.

POST

/sentiment-analysis Sentiment Analysis for Text

Schemas

Argument >

HTTPValidationError >

ValidationError >

Kodlama