BG-TEK Staj 2.Hafta

Model Eğitim Öncesi Features Analizi

Model eğitiminden önceki aşamalardan biri olan özellik(feature) belirleme aşaması üzerine çalışıldı. Genel olarak verilen yönergelerden yola çıkılarak python üzerinden yazılan bir betik ile bizimle paylaşılan kategori değerlerini kullanarak google arama motorundan ilk 100 adet website verisi çekilip kaydedildi. Daha sonra bu websitelerin kategorizasyonunun doğru olarak yapılabilmesi için özellikler(features) üzerine çalışıldı ve şimdilik sayıca ve nitelik olarak yetersiz kalsa da şu feature değerleri belirlendi :

-> SSL sertifikası var mı yok mu ?

SSL sertifikası daha profesyonel ve güvenilir sitelerde bulunur.

Örneğin, finans, sağlık veya haber siteleri genellikle SSL kullanır.

Kişisel bloglar veya daha düşük güvenlikli sitelerde SSL olmayabilir.

-> Page Load Time : -- Bizim hızımıza bağlı ise işlevsiz bir feature--

Daha hızlı yükleme süresi genellikle daha optimize edilmiş ve profesyonel sitelerle ilişkilendirilebilir (örneğin, haber veya sosyal medya siteleri). Daha yavaş yükleme süreleri ise daha ağır içeriklere sahip sitelerde veya düşük performanslı sunucularda görülür.

-> Page Length(Sayfa Uzunluğu) :

Uzun içerikli sayfalar genellikle bloglar, haber siteleri veya eğitimle ilgili sitelerde görülür. Daha kısa içerikli sayfalar ise e-ticaret, sosyal medya veya oyun sitelerinde olabilir.

-> Number of Links :

Daha fazla bağlantı genellikle haber siteleri, bilgi kaynakları veya bloglarda görülür. Az bağlantı ise kişisel bloglar veya tek sayfalık sitelerde yaygındır.

-> Number of Images :

Yüksek sayıda resim genellikle alışveriş, yemek tarifleri, moda ve görsel içerik odaklı sitelerde bulunur. Düşük resim sayısı ise daha çok metin odaklı içerik sunan sitelerde olabilir.

-> Subdomain : -- ayırt edici değil, kullanılmayacak--

Alt alan adı genellikle büyük şirketler, haber siteleri veya blog servis sağlayıcıları tarafından kullanılır. Kişisel siteler veya küçük bloglarda bu daha az yaygındır.

-> Reklam varlığı : -- Sağlıklı sonuçlar alamadım, betik bu feature için tekrardan yazılmalı--

Reklamlar genellikle haber siteleri, bloglar veya eğlence sitelerinde bulunabilir. Kurumsal veya resmi sitelerde reklamlar genellikle bulunmaz.

-> Number of videos :

Yüksek sayıda video genellikle dizi&film platformlarında, yetişkin içerik sitelerinde veya haber sitelerinde bulunabilir.

-> Hosting organization : -- Model eğitimi için gerekli olmayan bir feature, kullanılmayacak--

1.Hafta toplantısında dile getirildiği için üzerine durdum yalnız bu feature websitelerin kategorizasyonu için gerekli olmayabilir.

Bu özelliklerin(features) elde edilebilmesi için yazdığım betik ile bazı özelliklerin kullanılmamasının daha doğru olduğu sonucuna vardım ve bu durumlar beraber çıkardığım feature sayılarının bir model eğitimi için çok yetersiz kalacağını düşünüyorum. Öncelikle SSL sertifikası birçok websitede bulunan ve bu websiteleri kategorize edebilmemizde çok da kilit bir rol oynamayacak bir feature olabilir. Bu feature tüm websiteler için veri çekildikten sonra tekrardan değerlendirilmeli. Page Load Time değeri client bazlı bir veri sunuyorsa yanıltıcı bir değer olabilir. Aynı zamanda bir websitenin hızlı yüklenip yüklenmemesi ayırt edici bir özellik olmayabilir. Bazı verileri çektikten sonra Subdomain olup olmadığının kontrolünün model eğitimi için anlamlı bir feature olmayacağına karar verip listeden çıkardım. Reklam varlığı kurumsal veya resmi sitelerde kullanılamayacağından bu kategoriler özelinde model eğitimi için önemli bir feature olabilir yalnız yazdığım betik üzerinden bu feature için elde ettiğim veriler doğru sonuçları göstermiyordu. Bu feature için betiğin geliştirilmesi gerek.

Betikler : sırasıyla,

- ➔ Her kategori değeri için 100 website kaydeden çalışma
- ➔ Bu websiteleri için özellikleri(features) csv dosyasına kaydeden çalışma

```python
import csv
import os
from serpapi import GoogleSearch
from dotenv import load_dotenv

load_dotenv()
api_key = os.getenv('SERPAPI_KEY')


def search_google(query, num_results=100):
    params = {
        "q": query,
        "num": num_results,
        "engine": "google",
        "api_key": api_key
    }

    search = GoogleSearch(params)
    results = search.get_dict()
    return results.get('organic_results', [])


def save_to_csv(results, query, file_name="results.csv"):
    # CSV dosyasına yazmak için alanları belirleme
    fields = ['link', 'query'] # fields = ['position', 'title', 'link', 'snippet', 'query']

    file_exists = os.path.isfile(file_name)

    with open(file_name, mode='w', newline='', encoding='utf-8') as file:
        writer = csv.DictWriter(file, fieldnames=fields)

        if not file_exists:
            writer.writeheader()

        for result in results:
            link = result.get('link')
            # position: result.get('position'),
```

```
38              # title: result.get('title'),
39              # snippet: result.get('snippet')
40
41              writer.writerow({
42                  'link': link,
43                  # 'position': position,
44                  # 'title': title,
45                  # 'snippet': snippet
46                  'query': query
47              })
48
49
50  if __name__ == "__main__":
51      queries = [
52          "abortion", "advertising", "
    advocacy_organizations", "alcohol", "
    alternative_beliefs",
53          "armed_forces", "arts_and_culture", "auction"
    , "brokerage_and_trading", "business",
54          "child_abuse", "child_education", "
    content_servers", "dating", "denied_by_btk",
55          "digital_postcards", "discrimination", "
    domain_parking", "drug_abuse", "dynamic_content",
56          "dynamic_dns", "education", "entertainment",
    "explicit_violence", "file_sharing_and_storage",
57          "finance_and_banking", "folklore", "
    freeware_and_software_downloads", "gambling", "games"
    ,
58          "general_organizations", "global_religion", "
    government_and_legal_organizations", "hacking",
59          "health_and_wellness", "illegal_or_unethical"
    , "information_and_computer_security",
60          "information_technology", "instant_messaging"
    , "internet_radio_and_tv", "internet_telephony",
61          "job_search", "lingerie_and_swimsuit", "
    malicious_websites", "marijuana", "
    meaningless_content",
62          "medicine", "news_and_media", "
    newsgroups_and_message_boards", "nudity_and_risque",
63          "other_adult_materials", "
    peer_to_peer_file_sharing", "personal_vehicles", "
```

```
63 personal_websites_and_blogs",
64         "phishing", "plagiarism", "
   political_organizations", "pornography", "
   proxy_avoidance", "real_estate",
65         "reference", "restaurant_and_dining", "
   search_engines_and_portals", "secure_websites", "
   sex_education",
66         "shopping", "social_networking", "
   society_and_lifestyles", "spam_urls", "sports", "
   sports_hunting_and_war",
67         "streaming_media_and_download", "tobacco", "
   travel", "unrated", "weapons_sales", "
   web_based_applications",
68         "web_based_email", "web_chat", "web_hosting"
69     ]
70
71     # Her bir kategori için sorgu ve kayıt alma
72     for query in queries:
73         print(f"Searching for: {query}")
74         search_results = search_google(query)
75         save_to_csv(search_results, query)
76         print(f"{len(search_results)} results for '{
   query}' saved to CSV.")
77
```

```python
import csv
import time
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
import socket
import certifi


def has_ssl(url):
    try:
        response = requests.get(url, timeout=5, verify=certifi.where())
        return response.url.startswith("https")
    except requests.exceptions.RequestException:
        return None

# Burda kontrol ettiğimiz load time değeri bizim hızımıza bağlı olarak yanıltıcı olabilir mi ?
# def get_page_load_time(url, precision=2):
#     driver = webdriver.Chrome()
#     start_time = time.time()
#     driver.get(url)
#     load_time = time.time() - start_time
#     driver.quit()
#     return round(load_time, precision)


def get_page_length_and_links(url):
    try:
        response = requests.get(url, verify=certifi.where())
        soup = BeautifulSoup(response.text, 'html.parser')
        page_length = len(soup.get_text())
        link_count = len(soup.find_all('a'))
        return page_length, link_count
    except Exception:
        return None, None


```

```
38 def get_image_count(url):
39     try:
40         response = requests.get(url, verify=certifi.
   where())
41         soup = BeautifulSoup(response.text, 'html.
   parser')
42         image_count = len(soup.find_all('img'))
43         return image_count
44     except Exception:
45         return None
46 
47 
48 def get_video_count(url):
49     try:
50         response = requests.get(url, verify=certifi.
   where())
51         soup = BeautifulSoup(response.text, 'html.
   parser')
52         video_count = len(soup.find_all('video')) +
   len(soup.find_all('iframe'))
53         return video_count
54     except Exception:
55         return None
56 
57 
58 def get_ip_address(domain):
59     try:
60         ip_address = socket.gethostbyname(domain)
61         return ip_address
62     except socket.gaierror:
63         return None
64 
65 
66 def get_hosting_info(ip_address, retries=2, wait_time
   =2):
67     attempt = 0
68     while attempt < retries:
69         try:
70             response = requests.get(f"https://ipinfo.
   io/{ip_address}/json", verify=certifi.where())
71             data = response.json()
```

```
72              org_info = data.get('org')
73
74              # her hosting sağlayıcısının kendisine atanan ve AS ile başlayan bir değeri bulunur.
75              if org_info and org_info.startswith("AS"):
76                  # "AS" ile başlayan kısmı silip sadece numarayı alıyoruz ki model eğitiminde kullanabilelim.
77                  asn_number = org_info.split()[0][2:]  # örnek : "AS16509" -> "16509"
78                  return asn_number
79              return None
80          except requests.exceptions.ConnectionError as e:
81              print(f"Connection error: {e}. Retrying ...")
82              attempt += 1
83              time.sleep(wait_time)
84          except Exception as e:
85              print(f"Error fetching hosting info: {e}")
86              return None
87      return None
88
89
90  # org, gov gibi siteler reklam barındırmayacağından reklam varlığı bir feature olabilir yalnız betik ile  aldığım sonuçlar hatalı geldi.
91  # def has_ads(url):
92  #     try:
93  #         response = requests.get(url)
94  #         soup = BeautifulSoup(response.text, 'html.parser')
95  #
96  #         # Reklamlarla ilgili yaygın kullanılan HTML etiketlerini kontrol ediyoruz
97  #         ad_keywords = ['ad', 'ads', 'advertisement', 'ad-banner']
98  #
99  #         # 1. Iframe reklam kontrolü
```

```
100 #         iframes = soup.find_all('iframe')
101 #         if any('ad' in iframe.get('src', '').lower
    () for iframe in iframes):
102 #             return True
103 #
104 #         # 2. Yaygın kullanılan reklam sınıflarını
    kontrol ediyoruz
105 #         for keyword in ad_keywords:
106 #             if soup.find_all(class_=lambda
    class_name: class_name and keyword in class_name.
    lower()):
107 #                 return True
108 #
109 #         return False
110 #     except Exception as e:
111 #         print(f"Error checking ads: {e}")
112 #         return False
113
114
115 def update_csv(file_name="results_3.csv"):
116     # CSV dosyasını okuma ve yeni sütunlar(Features
    ) ekleme.
117     with open(file_name, mode='r', newline='',
    encoding='utf-8') as file:
118         reader = csv.DictReader(file)
119         rows = list(reader)
120
121     # Yeni sütunlar(features) ekleme
122     fieldnames = reader.fieldnames + ['
    ssl_certificate', 'page_length', 'link_count', '
    image_count',
123                                       'video_count'
    , 'hosting_org']
124
125     with open(file_name, mode='w', newline='',
    encoding='utf-8') as file:
126         writer = csv.DictWriter(file, fieldnames=
    fieldnames)
127         writer.writeheader()
128
129         for row in rows:
```

```
130             url = row['link']
131             domain = url.split("//")[-1].split("/")[
    0]  # URL'den domain'i çıkarma
132 
133             # IP ve hosting bilgilerini her website
    için kullanıp hosting_org değerini alma.
134             ip_address = get_ip_address(domain)
135             if ip_address:
136                 hosting_org = get_hosting_info(
    ip_address)
137             else:
138                 hosting_org = None
139 
140             # Features yazılması
141             ssl_status = has_ssl(url)
142             if ssl_status is not None:
143                 row['ssl_certificate'] = int(
    ssl_status)  # True veya False durumunda 1 veya 0
    yazılır
144             else:
145                 row['ssl_certificate'] = "N/A"  #
    SSL kontrolü yapılamadıysa "N/A" yazılır
146             # row['page_load_time'] =
    get_page_load_time(url)
147             row['page_length'], row['link_count'] =
    get_page_length_and_links(url)
148             row['image_count'] = get_image_count(url
    )
149             row['video_count'] = get_video_count(url
    )
150             row['hosting_org'] = hosting_org
151             # row['has_ads'] = int(has_ads(url))  #
    Sağlıklı çalışmadı
152 
153             # CSV yazım
154             writer.writerow(row)
155 
156 
157 if __name__ == "__main__":
158     update_csv()
159 
```