UNIVERSITATEA DIN
BUCUREȘTI

FACULTATEA DE
MATEMATICĂ ȘI
INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

INSTAGRAM ANALYTICS

Absolvent
Buhai Darius

Coordonator științific
Dumitran Adrian-Marius

București, iunie 2022

Rezumat

Creșterea numărului de like-uri primite în postările de Instagram, cât și analiza asupra urmăritorilor activi reprezintă niște ustensile destul de importante pentru orice influencer de success.

Pentru aceasta vom crea o aplicație mobilă ce va oferi predicții asupra numărului de like-uri pe care le poate primi o postare pe Instagram. Pentru o predicție cât mai reală voi antrena un model Secvențial cu ~200.000 de postări de Instagram extrase folosind api-ul oferit de Facebook [1] și implementat în limbajul python [2] [3].

Aplicația de mobil va fi realizată în Flutter [4] și va oferi o interfață grafică pentru prezicerea aprecierilor în funcție de descriere, data, ora și imagine. De asemenea, vom extrage informații relevante din contul utilizatorului de Instagram, ce vor fi folosite în predicția noastră (număr de urmăritori și media like-urilor).

Abstract

As an influencer, growing the number of likes for your Instagram posts and analyzing the number of active followers for your page should be a priority, and using various tools for this can become very helpful.

For that, we will create a mobile app that will predict the number of likes that an Instagram post could receive. For the prediction to work, we will train a Sequential model on ~200.000 Instagram posts which will be extracted using the API offered by Facebook [1] and implemented in python [2] [3].

The mobile app will be created in Flutter [4] and it will include a graphic interface for predicting the number of likes based on a given description, date-time, and image. Also, we will retrieve relevant insights from the user's Instagram profile which will be needed for our prediction (followers and mean likes).

Cuprins

Capitolul 1

Introducere

1.1 Motivație

Analiza audienței și a preferințelor oamenilor pe internet a fost dintotdeauna un subiect de interes pentru mine. Astfel prin această lucrare mă voi axa pe predicția aprecierilor postărilor de instagram în funcție de caracteristicile cele mai importante (câte persoane apar în imagine, descrierea pusă, ora și ziua în care e postată).

Deoarece pentru orice predicție bună avem nevoie de un set de date cât mai mare, voi folosi postări reale extrase folosind api-ul oferit de Facebook [1] și implementat în Python [2] [3], de la top 300 cei mai urmăriți influenceri.

1.2 Domenii abordate

Această lucrare va include următoarele 3 domenii principale:

- **Inteligența Artificială**, folosită pentru realizarea modelelor de tip regression, antrenarea lor și analiza metricilor.
- **Natural Language Processing**, folosit pentru a extrage caracteristicile din descrierile postărilor.
- **Aplicații Mobile**, realizând o aplicație mobilă în **Flutter** [4] ce rulează atât pe iOS cât și pe Android.

1.3 Structura lucrării

Lucrarea de licență este impărțită în 2 capitole principale:

- **Predicția like-urilor**: Include extragerea dataset-ului, preprocesarea, analiza datelor, extragerea caracteristicilor, definirea modelului și antrenarea sa.

- **Aplicația mobilă**: Descrierea procesul de creare a aplicației mobile și a backend-ului ce oferă o interfață vizuală pentru modelul definit.

1.3.1 Predicția like-urilor

Pentru a realiza o predicție reală asupra numărului de like-uri ce pot fi obținute de o postare, vom aduna cât mai multe postări de la top 300 influenceri și vom antrena un model Secvențial pe diversele caracteristici ce alcătuiesc o postare de success (cuvintele folosite în descriere, numărul de persoane din imagine, numărul de zâmbete din imagine, ziua în care care a fost postată, etc.).

1.3.2 Aplicația mobilă

Aplicația mobilă va fi realizată în Flutter și va include 3 pagini principale:

1. **Pagina de prezicere a numărului de like-uri**, aici utilizatorul poate încărca imaginea, ziua și ora la care dorește să o posteze cât și descrierea. Folosind modelul antrenat anterior, utilizatorul va primi o predicție asupra numărului de like-uri raportat la numărul de followeri pe care îi are.

2. **Pagina de asociere cu contul de Instagram**. Aici utilizatorul își va completa username-ul de instagram și va primii analize asupra numărului de urmăritori și media like-urilor postărilor. Aceste metrici vor fi utilizate automat în predicția postărilor.

3. **Pagina de setări**, unde utilizatorul își va putea administra contul și setările aplicației.

Capitolul 2

Predicția like-urilor

2.1 Preluarea setului de date

2.1.1 Conturile de instagram

În prima faza, vom avea nevoie de conturile de instagram din care să preluăm postările, pentru această am folosit dataset-ul oferit de **Kaggle**: top_1000_instagram_influencers [16]. Acest dataset ne oferă top 1000 cei mai urmăriți influenceri de pe instagram în ordine descrescătoare (după numărul de urmăritori), alături de username-ul și numărul lor de urmăritori.

	Rank	Account	Title	Link	Category	Fol...	Audie...	Authenti
1	1	cristiano	Cristiano Ronaldo	https://www.instagram.com/cristiano/	Sports with a ball	400100000.0	India	
2	2	kyliejenner	Kylie ♥	https://www.instagram.com/kyliejenne…	Fashion|Modeling|Beauty	308800000.0	United States	
3	3	leomessi	Leo Messi	https://www.instagram.com/leomessi/	Sports with a ball|Family	306300000.0	Argentina	
4	4	kendalljenner	Kendall	https://www.instagram.com/kendalljen…	Modeling|Fashion	217800000.0	United States	
5	5	selenagomez	Selena Gomez	https://www.instagram.com/selenagome…	Music|Lifestyle	295800000.0	United States	
6	6	zendaya	Zendaya	https://www.instagram.com/zendaya/	Cinema|Actors/actresses|Fas…	127800000.0	United States	
7	7	kimkardashian	Kim Kardashian W…	https://www.instagram.com/kimkardash…	Fashion|Beauty	284900000.0	United States	
8	8	beyonce	Beyoncé	https://www.instagram.com/beyonce/	Music|Fashion	237200000.0	United States	
9	9	arianagrande	Ariana Grande	https://www.instagram.com/arianagran…	Music	294100000.0	United States	
10	10	billieeilish	BILLIE EILISH	https://www.instagram.com/billieeili…	Music	100000000.0	United States	
11	11	neymarjr	NJ	https://www.instagram.com/neymarjr/	Sports with a ball	169800000.0	Brazil	
12	12	lalalalisa_m	LISA	https://www.instagram.com/lalalalisa…	Music	73100000.0	Indonesia	
13	13	jennierubyjane	J	https://www.instagram.com/jennieruby…	Music	62300000.0	Indonesia	
14	14	tomholland2013	Tom Holland	https://www.instagram.com/tomholland…	Cinema|Actors/actresses	60900000.0	United States	
15	15	virat.kohli	Virat Kohli	https://www.instagram.com/virat.kohl…	Sports with a ball	182600000.0	India	
16	16	abcdefghi__lmn…	Jungkook	https://www.instagram.com/abcdefghi_…	<null>	32100000.0	United States	
17	17	khaby00	Khaby Lame	https://www.instagram.com/khaby00/	<null>	69200000.0	Iran	
18	18	khloekardashian	Khloé Kardashian	https://www.instagram.com/khloekarda…	Clothing|Outfits|Lifestyle	219400000.0	United States	
19	19	sooyaaa__	JISOO♥	https://www.instagram.com/sooyaaa__/	Music|Beauty	56100000.0	Indonesia	
20	20	justinbieber	Justin Bieber	https://www.instagram.com/justinbieb…	Music	219800000.0	India	
21	21	thv	V	https://www.instagram.com/thv/	<null>	34100000.0	<null>	
22	22	roses_are_rosie	ROSÉ	https://www.instagram.com/roses_are_…	Music	55100000.0	Indonesia	
23	23	badgalriri	badgalriri	https://www.instagram.com/badgalriri/	Music|Beauty	120900000.0	United States	
24	24	nickiminaj	Barbie	https://www.instagram.com/nickiminaj/	Music	174000000.0	United States	
25	25	iamcardib	Cardi B	https://www.instagram.com/iamcardib/	Music|Fashion	123100000.0	United States	
26	26	jin	Jin of BTS	https://www.instagram.com/jin/	<null>	28400000.0	United States	
27	27	agustd	SUGA of BTS 민윤기	https://www.instagram.com/agustd/	<null>	27900000.0	<null>	
28	28	therock	therock	https://www.instagram.com/therock/	Cinema|Actors/actresses|Fit…	295800000.0	India	
29	29	instagram	Instagram	https://www.instagram.com/instagram/	Photography	469600000.0	India	

Figura 2.1: Conturile de instagram

2.1.2 Preluarea postărilor

Pentru a prelua postările fiecărui cont în parte, vom folosi librăria instagram_private_api [2], ce apelează api-ul oficial Facebook [1].

O problemă destul de importantă întâmpinată a fost limitarea instagramului asupra numărului de postări publice ce pot fi preluate în același timp. Pentru a trece de la 64 de postări pe cont la 2.000 de postări pe cont, am folosit autentificarea cu un cont de instagram. Cu toate acestea, chiar și autentificat, instagram permite ~100 de request-uri la câteva minute, de aceea m-am limitat la 300 de conturi, în total obținând ~272.407 de postări. [1]

Algoritmul nostru se autentifică la un cont de instagram privat și salvează sesiunea de autentificare într-un fișier în format JSON. După aceea, pentru fiecare cont în parte, folosind token-ul de autentificare salvat, se conectează la contul nostru și încarcă postările profilului de instagram interogat. Fiecare request către api-ul instagram returnează ~64 de postări cât și un token ce ne redirecționează către următorul set de postări (următoarele 64 etc.). Pentru a nu supra-solicita api-ul instagram, am preluat maximum 2000 de postări pe cont (aproximativ 31 de request-uri).

```
# Get instagram posts using private_instagram_api
def getInstagramPosts(self, ig_username, limit_posts=64):
    ig_posts = list()
    last_max_id = None
    print(f"Loading @{ig_username}: [", end="")
    while limit_posts > 0:
        try:
            user_feed_info = self.api.username_feed(user_name=ig_username,
            count=64, max_id=last_max_id)
            current_posts = user_feed_info['items']
            for post in current_posts:
                post['description'] = ""
                if post['caption'] is not None and 'text' in post['caption']:
                    post['description'] = post['caption']['text']
            ig_posts.extend(current_posts)
            limit_posts -= user_feed_info['num_results']
            if not user_feed_info['more_available']:
                break
            if 'next_max_id' not in user_feed_info:
                break
            last_max_id = user_feed_info['next_max_id']
            print("#", end="")
        except Exception as e:
            print(e)
            return ig_posts
    print("]")
```

```
    return ig_posts
```

Funcția de mai sus este apelată de următoarea funcție în mod repetat, pentru fiecare cont de instagram în parte (din lista extrasă anterior). Pentru fiecare cont vom salva postările într-un fișier separat, păstrând doar datele care ne interesează.

```
# Save posts to csv file. Keep only relevant informations
def saveInstagramAccountsPosts(self):
    accounts = pd.read_csv(self.ACCOUNTS_FILE)
    for idx, account in accounts.iterrows():
        should_repeat = True
        while should_repeat:
            should_repeat = False
            try:
                username = account['Account']
                filepath = f"{self.CURRENT_POSTS_PATH}@{username}.csv"
                if os.path.exists(filepath):
                    continue
                posts = self.getInstagramPosts(username, limit_posts=2000)
                if len(posts) == 0:
                    print("No posts found")
                    return
                posts_pd = pd.DataFrame(posts)
                posts_pd = posts_pd[posts_pd.columns[posts_pd.columns.isin([
                    'id', 'taken_at', 'media_type',
                    'is_unified_video',
                    'is_paid_partnership','next_max_id',
                    'comment_count','like_count',
                    'title', 'link', 'images', 'description', 'original_width',
                    'original_height'])]]

                posts_pd.to_csv(filepath)
                print(f"Saved {username} to {filepath}")
            except Exception as e:
                print(e)
                return
```

După rularea scriptului nostru, vom obține pentru fiecare cont in parte următorul output ce descrie pașii rulați de algoritm:

Figura 2.2: Extragerea datelor

2.1.3 Preluarea imaginilor

Ținând cont că avem access la peste 270.000 de postări, unde fiecare imagine are aproximativ 10Kb, un calcul rapid ne spune că am consuma peste 20GB pentru a salva toate imaginile. Pentru a nu incărca spațiul degeaba, vom parcurge fiecare postare în parte, îi vom salva imaginea, vom analiza și salva caracteristicile acesteia, iar apoi o vom șterge. Caracteristicile extrase sunt explicate mai în detaliu în secțiunea 2.3.1.

2.2 Preprocesarea datelor

De cele mai multe ori descrierile regăsite pe internet nu sunt într-o formă optimă pentru a fi procesate și a se aplica metode de învățare. Devine importantă **normalizarea** textului prin aplicarea unei serii de pași de preprocesare. Am aplicat un set de pași de preprocesare pentru a-l face potrivit pentru algoritmii de învățare și pentru a reduce dimensiunea setului de caracteristici. [15]

Din punctul de vedere al clasificării unui text, cele mai importante aspecte ale preprocesării necesare setului de date ales implică:

- **Eliminarea userului:** Fiecare utilizator are un nume de utilizator unic. Orice lucru îndreptat către acel utilizator poate fi indicat scriind numele de utilizator precedat de „@”, care nu furnizează nicio informație utilă, fiind un nume propriu.

```
dataset['description'] = np.vectorize(Preprocessing.removePattern)
    (dataset['description'], "@[\w]*")
```

- **Eliminarea link-urilor:** Utilizatorii partajează adesea hyperlinkuri în postarile lor.

```
dataset['description'] = np.vectorize(Preprocessing.removePattern)(
    dataset['description'], "https?://[\S]+")
dataset['description'] = np.vectorize(Preprocessing.removePattern)(
    dataset['description'], "http?://[\S]+")
```

- **Eliminarea punctuațiilor, numerelor și a caracterelor speciale:** Utilizatorii folosesc punctuațiile, numerele și caractere speciale într-un mod abuziv, intenționat sau accidental, iar acestea nu impacteaza in mod direct numărul de like-uri primite pe postare

```
dataset['description'] = dataset['description'].str.replace("[^a-zA-Z#]", " ")
```

- **Eliminarea cuvintelor scurte:** Interjecțiile și cuvintele de legătura nu au un rol important în predicția postărilor, de aceea le vom elimina.

```
dataset['description'] = dataset['description'].apply(
    lambda x: ' '.join([w for w in x.split() if len(w) > 3]))
```

- **Modificarea literelor mari în litere mici:** Ajută la menținerea fluxului de consistență și extragerii de text.

```
dataset['description'] = dataset['description'].apply(
    lambda x: ' '.join([x.lower()]))
```

- **Eliminarea caracterelor repetitive:** În limbajul de zi cu zi, oamenii de multe ori nu sunt strict gramaticali. Vor scrie lucruri precum „I looooooove it”, pentru a sublinia cuvântul dragoste. Cu toate acestea, computerele nu știu că „looooooove” este o variație a „iubirii” decât dacă li se spune.

```
dataset['description'] = dataset['description'].apply(lambda x: ' '
    .join([re.sub(r'(.)\1+', r'\1\1', x)]))
```

- **Eliminarea spațiilor multiple:** De cele mai multe ori, textele conțin spații suplimentare sau în timpul efectuării tehnicilor de preprocesare de mai sus, rămâne mai mult de un spațiu intre cuvinte.

```
dataset['description'] = dataset['description'].apply(
    lambda x: ' '.join([re.sub(r'\s+', ' ', x)]))
```

- **Eliminarea hashtag-urilor:** Este destul de comun să folosim hashtag-uri în postările de instagram pentru a sublinia cuvintele cheie din postare. De aceea vom transforma cuvintele de forma '#word' în 'word'.

```
dataset['description'] = dataset['description'].apply(lambda x: ' '
    .join([re.sub(r'#(\S+)', r' \1 ', x)]))
```

- **Stemming:** Există multe variante de cuvinte care nu aduc informații noi și creează redundanță, aducând în cele din urmă ambiguitate atunci când antrenăm modele de învățare automată pentru predicții. De aceea vom folosi funcția PorterStemmer pentru a transforma cuvintele în token-uri.

```
tokenized_test = dataset['description'].apply(lambda x: x.split())
# normalize the tokenized descriptions.
stemmer = PorterStemmer()
tokenized_test = tokenized_test.apply(
    lambda x: [stemmer.stem(j) for j in x])  # stemming
try:
    for i in range(len(tokenized_test)):
        tokenized_test[i] = ' '.join(tokenized_test[i])
    dataset['description'] = tokenized_test
except Exception as e:
    print(e.__str__())
```

În final, vom obține pentru fiecare descriere în parte mai multe token-uri extrase. Token-urile sunt reprezentări simplificate și generalizate ale cuvintelor. Token-ul de forma 'thi' reprezintă cuvântul 'this', iar token-ul 'love' poate reprezenta toate formele acestui cuvânt, precum: 'loving', 'loved', 'loves' etc.

2.3 Extragerea caracteristicilor

2.3.1 Din imagini

Știm că o postare de success pe instagram conține cât mai multe persoane în ea. De aceea, vom extrage din imaginile aferente fiecărei postări, numărul de fețe din poză cât și câte dintre ele zâmbesc.

Pentru asta, am folosit librăria pypi.org/project/opencv-python/ [5], ce ne oferă modele predefinite de image processing, numite haarscascades. Am folosit 3 astfel de modele pentru a analiza imaginile și am extras pentru fiecare postare numărul de fețe și zâmbete din ele.

```
def getImageFeatures(image_path) -> Features:
    # Read the image
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faceCascade = cv2.CascadeClassifier("haarscascades/frontalface.xml")
    smileCascade = cv2.CascadeClassifier("haarscascades/smile.xml")
    eyesCascade = cv2.CascadeClassifier("haarscascades/eye.xml")
    persons: list[Person] = list()
    # Detect faces in the image
    faces = faceCascade.detectMultiScale(gray, scaleFactor=1.2,
    minNeighbors=5, minSize=(30, 30))
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
        face_cropped_gray = gray[y:y + h, x:x + w]
        # Detect smiles in the image
        smiles = smileCascade.detectMultiScale(face_cropped_gray, scaleFactor=1.2)
        # Detect eyes in the image
        eyes = eyesCascade.detectMultiScale(face_cropped_gray, scaleFactor=2)
        persons.append(Person(smiles=len(smiles) > 0, eyes=max(2, len(eyes))))
    return Features(persons=persons)
```

Ca și exemplu, putem vedea cum din următoarea imagine au fost extrase numărul de persoane din poză (modelul nu este 100% eficient, de aceeea unele persoane nu sunt detectate, în special cele care nu se uită direct spre cameră).

Figura 2.3: Exemplu detecție persoane

2.3.2 Din descrieri

Pentru a include descrierile postărilor în modelul nostru, le vom transforma în reprezentări numerice vectoriale.

Astfel, vom crea un vocabular folosind toate descrierile din dataset-ul nostru, iar apoi pentru fiecare postare în parte vom crea un set de caracteristici extrase din acest vocabular.

```
def extractDatasetDescriptionFeatures(self, total_features=400) -> Tensor:
    # Create vocabulary
    self.createVocab()
    print("Created vocabulary")
    # For each post, create vectorize
    return self.createVectorize(total_features)
```

Pentru a construi vocabularul nostru de cuvinte, vom aduna toate token-urile utilizate în descrieri (preprocesate în capitolul anterior), iar apoi folosind funcția Counter din collections le vom grupa într-un dicționar după numărul de utilizări a fiecărui cuvânt în parte (cheie = cuvânt, valoare = număr de utilizări).

```
@staticmethod
def wordFrequency(data: pd.Series, min_occurrences):
    all_words = [word.lower() for sentence in data for word in
    sentence.split(' ')]
    sorted_vocab = sorted(dict(Counter(all_words)).items(),
```

```
    key=operator.itemgetter(1), reverse=True)
    return [k for k, v in sorted_vocab if v > min_occurrences]
```

După aceea vom elimina toate cuvintele de legătură și semnele de punctuație din vocabularul creat. În final, am obținut un vocabular cu peste 11.000 de token-uri, dintre care cele mai frecvente sunt:

- 'thi' (35,054 de utilizări), acest token reprezină cuvântul 'this'.
- 'love' (23,997 de utilizări). Forme incluse: 'loving', 'loved', 'lover' etc.
- 'thank' (18,295 de utilizări). Forme incluse: 'thanks', 'thank you' etc.
- 'happi' (11,380 de utilizări). Forme incluse: 'happy', 'happier' etc.
- 'time' (10,395 de utilizări). Forme incluse: 'times', 'timer' etc.
- 'link' (9,338 de utilizări). Forme incluse: 'links' etc.

```
def removePunctuationAndStopwords(self):
    words = []
    for word in self.vocab:
        if word not in nlp.Defaults.stop_words and word != ' ' and
        word not in string.punctuation:
            words.append(word)
    self.vocab = words

def createVocab(self):
    if self.prediction_mode:
        self.vocab = # [...]
        return
    self.vocab = self.wordFrequency(self.posts['description'], min_occurrences=18)
    self.removePunctuationAndStopwords()
    vocab = pd.DataFrame(self.vocab)
    vocab.to_csv(self.VOCAB_FILE)
```

Mai departe, folosind vocabularul creat, am păstrat top 200 dintre cele mai frecvente cuvinte, iar pentru fiecare postare în parte, am păstrat doar cuvintele regăsite în vocabular și folosind **Bag Of Words** [9] am obținut 200 de caracteristici noi.

```
@staticmethod
def pad(samples, max_length):
    return torch.tensor([
        sample[:max_length] + [0] * max(0, max_length - len(sample))
        for sample in samples
    ])
```

```
7
8   def createVectorize(self, total_features):
9       vectorized = []
10      for sentences in self.posts['description']:
11          current_vector = []
12          sentences_sep = sentences.split(" ")
13          for i in range(total_features):
14              word = self.vocab[i]
15              if word in sentences_sep:
16                  current_vector.append(1)
17              else:
18                  current_vector.append(0)
19          vectorized.append(current_vector)
20      return torch.FloatTensor(vectorized)
```

2.3.3 Din alte informații

O altă caracteristică importantă atunci când vine vorba de prezicerea like-urilor unei postări este reprezentată de ziua și ora la care este publicată. Astfel o postare publicată seara, după ora 17:00 (după finalizare programului de muncă) ar putea obține mai multe like-uri decât o postare publicată noaptea (când nimeni nu ar fi activ). De asemenea, zilele de weekend obțin în general audiențe mult mai crescute decât zilele din cursul săptămânii.

Astfel, pentru fiecare postare în parte am adăugat 13 caracteristici noi, reprezentate de ziua săptămânii în care a fost postată (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday) și intervalul orar (8:00-12:00, 12:00-17:00, 17:00-22:00, 22:00-2:00, 2:00-8:00).

```
1   # Day of week
2   available_dates = ['Sunday', 'Monday', 'Tuesday',
3   'Wednesday', 'Thursday', 'Friday', 'Saturday']
4   time_intervals = [(8, 12), (12, 17), (17, 22), (22, 2), (2, 8)]
5
6   for idx, post in self.posts.iterrows():
7       # Day posted
8       try:
9           date_posted = datetime.fromtimestamp(post['taken_at'])
10      except:
11          date_posted = datetime.today()
12      for date_day in available_dates:
13          new_post[date_day] = False
14      new_post[date_posted.strftime('%A')] = True
15
16      # Time interval
17      for time_interval in time_intervals:
```

```
        new_post[str(time_interval)] = False
        hour_start = int(date_posted.strftime('%H'))
        if time_interval[0] < hour_start <= time_interval[1]:
            new_post[str(time_interval)] = True

    # [...]
```

Pe lângă acestea, am adăugat și media like-urilor raportate la fiecare cont în parte, raportul dintre like-uri și media lor, raportul dintre like-uri și numărul de urmăritori. Acestea le vom folosi drept metrici.

```
mean_likes = dict()
if not self.prediction_mode:
    for idx, post in self.posts.iterrows():
        if type(post['likes']) is not float and
        type(post['likes']) is not int:
            continue
        if post['username'] in mean_likes:
            old_mean, old_count = mean_likes[post['username']]
            mean_likes[post['username']] = ((old_mean *
            old_count + post['likes']) / (old_count + 1), old_count + 1)
        else:
            mean_likes[post['username']] = (post['likes'], 1)

    for idx, post in self.posts.iterrows():
        if not self.prediction_mode:
            new_post['likes/followers'] = post['likes'] / post['followers']
            new_post['likes/mean'] = post['likes'] /
            mean_likes[post['username']][0]
            new_post['mean'] = mean_likes[post['username']][0]
```

2.3.4 Vizualizarea datelor

Combinând toate caracteristicile extrase, vom obține un fișier CSV cu ~272.407 de postări a câte 218 caracteristici. Caracteristicile sunt notate după cum urmează: [14] [19]

- Invervalele orare sunt notate cu paranteze rotunde (start, end).
- Cuvintele utilizate în descrieri sunt notate cu word: (cuvântul).
- Zilele săptămânii sunt notate în engleza

În final, setul nostru de date va arăta astfel:

	C4	C6	C2	C3	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25
1	followers	likes	smiles	faces	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	(8, 12)	(12, 17)	(17, 22)	(22, 2)	(2, 8)	word: thi	word: love	word: thank	word: happi	word: tir
2	400100000.0	2240988.0	<null>	<null>	False	False	True	False	False	False	False	False	False	True	False	False	0	0	0	0	0
3	400100000.0	7886085.0	0.0	1.0	False	False	True	False	False	False	False	False	False	False	False	False	1	0	1	0	0
4	400100000.0	19643288.0	<null>	<null>	False	False	False	False	False	False	True	True	False	False	False	False	0	1	0	0	0
5	400100000.0	8363211.0	0.0	0.0	False	False	False	False	True	False	False	False	False	False	False	False	0	0	1	0	0
6	400100000.0	11434276.0	0.0	1.0	False	False	False	False	False	False	True	False	False	True	False	False	0	0	0	0	0
7	400100000.0	18780508.0	<null>	<null>	False	False	False	False	True	False	False	False	False	True	False	False	1	1	1	0	1
8	400100000.0	12838882.0	<null>	<null>	False	False	False	False	True	False	False	False	True	False	False	False	1	0	1	0	0
9	400100000.0	16188034.0	0.0	0.0	False	True	False	False	False	False	False	False	False	True	False	False	0	0	0	0	0
10	400100000.0	9332713.0	0.0	1.0	True	False	False	False	False	False	False	False	True	False	False	False	0	0	0	0	1
11	400100000.0	10255392.0	0.0	0.0	False	False	False	False	False	False	True	False	False	True	False	False	1	0	0	1	0
12	400100000.0	8035532.0	0.0	2.0	False	False	False	True	False	False	False	False	False	True	False	False	0	0	0	0	0
13	400100000.0	7873294.0	<null>	<null>	False	False	False	False	False	False	True	False	False	False	False	False	1	1	0	0	0
14	400100000.0	7953689.0	2.0	2.0	False	False	False	False	True	False	False	False	False	False	False	False	0	0	0	0	0
15	400100000.0	6627776.0	0.0	0.0	False	False	False	False	True	False	False	False	True	False	False	False	0	0	1	0	0
16	400100000.0	6463310.0	0.0	1.0	False	False	False	True	False	False	False	False	False	True	False	False	0	0	0	0	0
17	400100000.0	5294226.0	1.0	1.0	False	False	True	False	False	False	False	False	False	True	False	False	0	0	1	0	0
18	400100000.0	11170867.0	0.0	0.0	True	False	False	False	False	False	False	False	True	False	False	False	0	0	0	0	0
19	400100000.0	2180145.0	0.0	1.0	False	False	False	False	True	False	False	False	False	True	False	False	0	0	0	0	0
20	400100000.0	10752159.0	<null>	<null>	False	False	False	True	False	False	False	False	False	True	False	False	0	0	0	0	0
21	400100000.0	9015981.0	0.0	4.0	False	False	False	True	False	False	False	True	False	False	False	False	0	0	0	0	0
22	400100000.0	9017854.0	0.0	0.0	False	False	False	True	False	False	False	False	False	False	False	False	0	0	0	0	0
23	400100000.0	8679568.0	<null>	<null>	False	True	False	False	False	False	False	False	False	True	False	False	0	0	0	0	0
24	400100000.0	6951233.0	0.0	1.0	False	False	False	False	False	False	True	False	False	True	False	False	0	0	0	0	0
25	400100000.0	4580145.0	<null>	<null>	False	False	False	False	False	False	True	True	False	False	False	False	0	0	0	0	0
26	400100000.0	8954391.0	0.0	0.0	False	False	False	False	False	True	False	False	False	True	False	False	0	0	0	0	0
27	400100000.0	7753999.0	<null>	<null>	False	False	False	False	False	True	False	False	False	False	False	False	0	0	0	0	0
28	400100000.0	5802663.0	<null>	<null>	False	False	False	True	False	False	False	False	False	True	False	False	0	0	0	0	0
29	400100000.0	9990516.0	1.0	1.0	False	True	False	False	False	False	False	False	False	False	False	False	0	0	0	0	0
30	400100000.0	2916382.0	0.0	1.0	False	True	False	False	False	False	False	True	False	False	False	False	1	0	1	0	0
31	400100000.0	10254770.0	1.0	1.0	False	False	False	False	False	False	True	False	False	True	False	False	0	1	0	1	0
32	400100000.0	7910983.0	1.0	1.0	False	True	False	False	False	False	False	False	False	True	False	False	1	0	0	0	0
33	400100000.0	2777085.0	0.0	1.0	False	True	False	False	False	False	False	False	True	False	False	False	0	0	0	0	1

Figura 2.4: Vizualizarea datelor

2.3.5 Importanța caracteristicilor extrase

Pentru început, am căutat o legătură între numărul de like-uri și urmăritori ai contului. Astfel am realizat următorul grafic ce descrie discrepanța dintre cele 2 metrici:

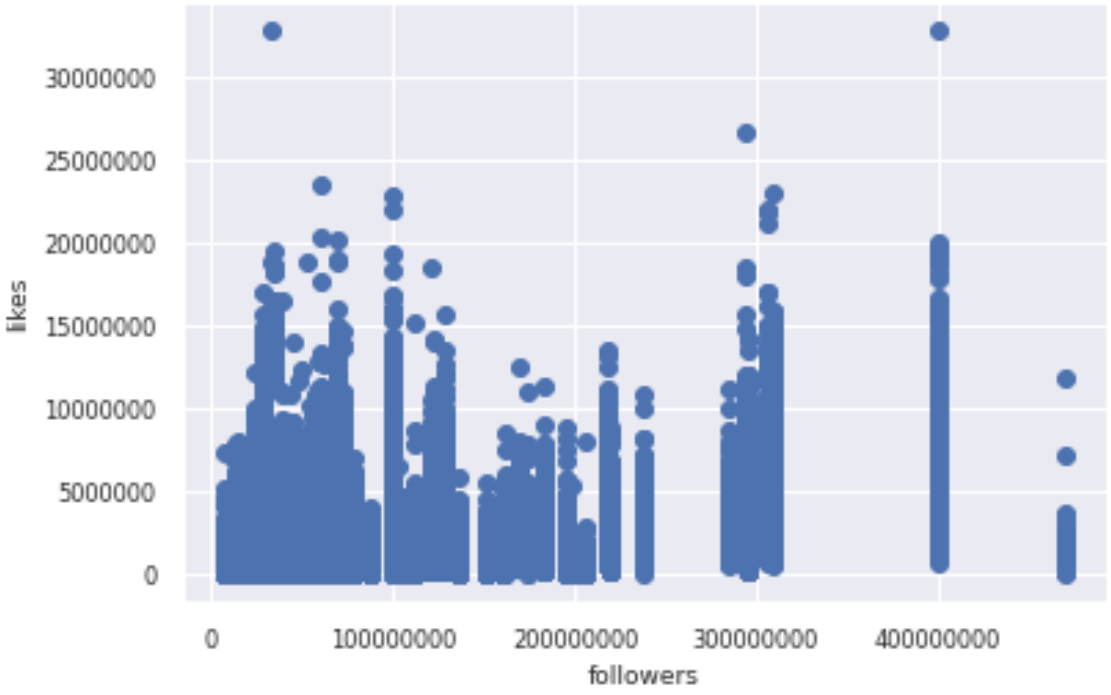

Figura 2.5: Followers/Likes

Din graficul prezentat putem observa că nu există întotdeauna o legătura directă între numărul de like-uri și numărul de urmăritori ai unui cont.

Pentru a observă legătură dintre caracteristicile extrase, vom crea o matrice de confuzie cu importanța legăturii dintre caracteristici, raportate la numărul de like-uri pe media lor (medie realizată pe fiecare cont în parte).

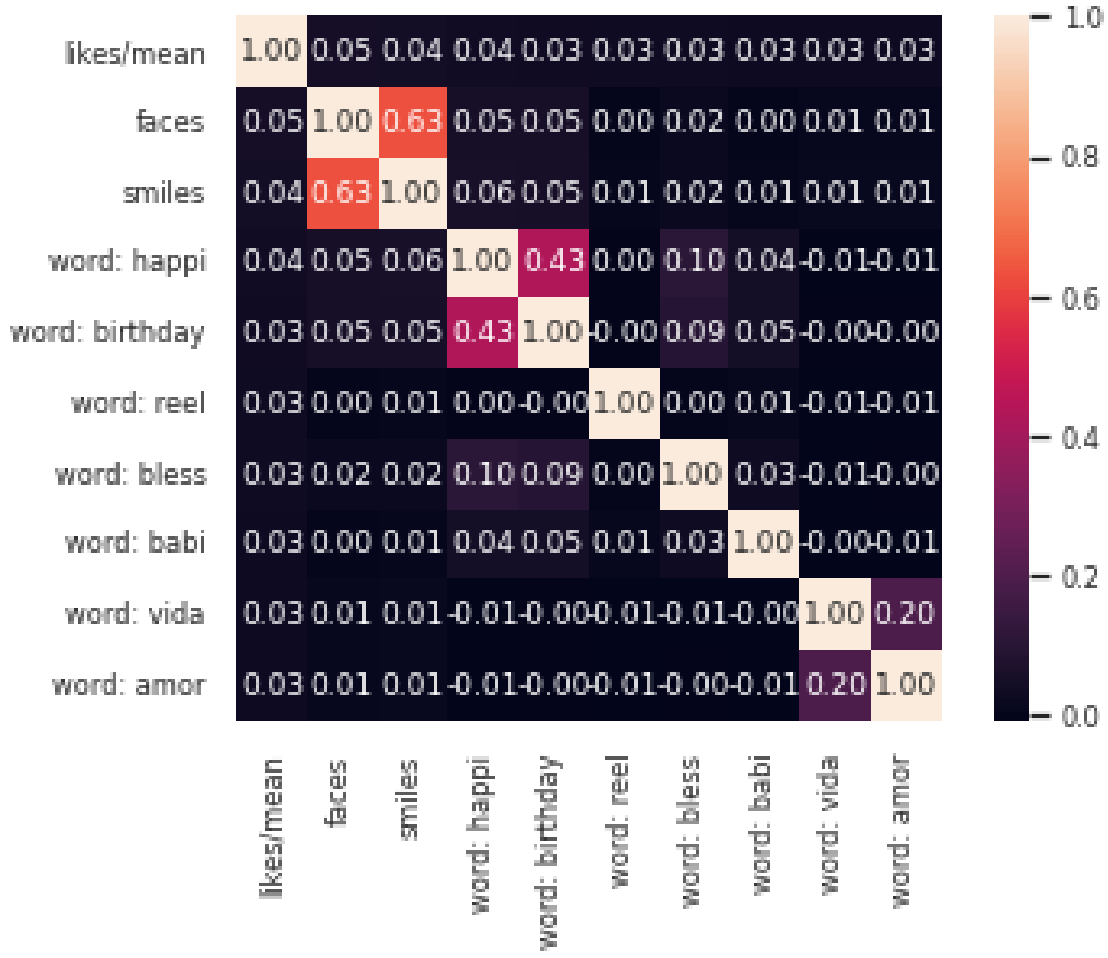

Figura 2.6: Importanța legăturii dintre statistici

De asemenea, avem următoarea distribuție dintre numărul de like-uri pe media like-urilor fiecărui cont. Acest grafic are punctul maxim la 1.7, având o amprentă totală de 5.

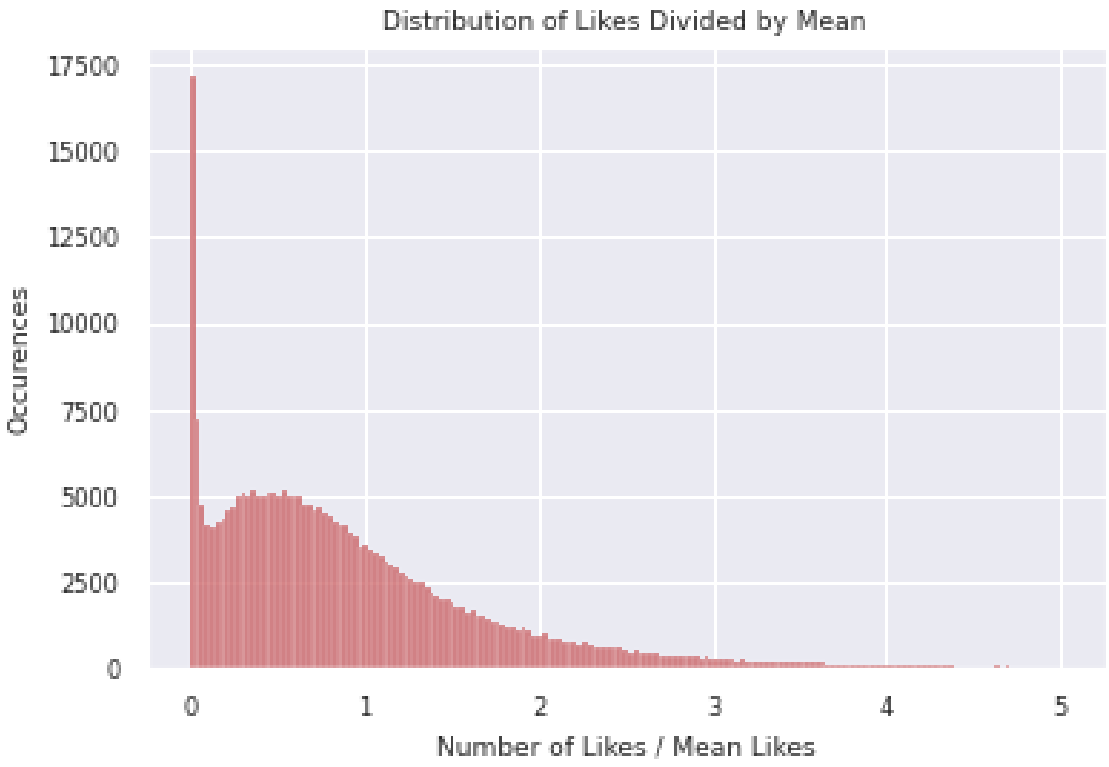

Figura 2.7: Distribuția like-urilor pe media lor

Dintre caracteristicile extrase, dorim să aflăm care sunt cele mai dominante și direct proporționale cu numărul de like-uri primite în postări. Astfel, am creat un set de antrenare ce conține caracteristicile extrase relevante (nu vom avea nevoie de followers, comments sau engagement, deoarece acestea sunt deja direct proporționale cu numărul de like-uri) și numărul de like-uri primite pe fiecare postare în parte. [11]

Folosind **Decision Tree Regression**, am obținut următoarele rezultate:

```
Feature: faces, importance: 0.03313992513678686
Feature: smiles, importance: 0.030565945767243096
Feature: (17, 22), importance: 0.026483121858356915
Feature: (12, 17), importance: 0.022830673612696174
Feature: Saturday, importance: 0.019839661195867844
Feature: Wednesday, importance: 0.019517213420801326
Feature: Tuesday, importance: 0.018437200399632286
Feature: Thursday, importance: 0.01838019771953706
Feature: (2, 8), importance: 0.017578033412455427
Feature: Monday, importance: 0.017526708567852874
Feature: Friday, importance: 0.01649033657799903
Feature: word: love, importance: 0.01625519712807099
Feature: (8, 12), importance: 0.016116062120101745
Feature: Sunday, importance: 0.015498441894218878
Feature: word: thi, importance: 0.014318924832485973
```

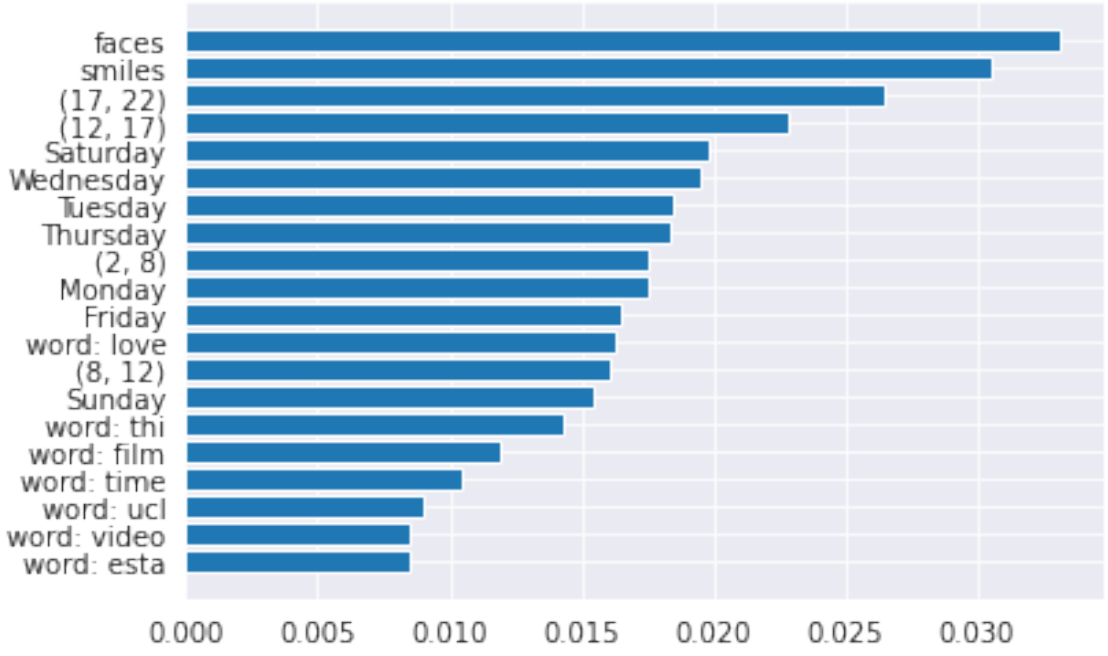

Figura 2.8: Importanța caracteristicilor - Decision Tree

Din acestea, putem observa cele mai importante caracteristici ce impactează numărul de like-uri primite în mod direct, precum:

- Numărul de fețe și zâmbete din imagini.
- Folosirea cuvântului 'love' sau alte forme ale acestuia, ex: 'loves', 'loving' etc.
- Orele cele mai bune de postat sunt între 17:00 și 22:00.
- Ziua cea mai bună în care să postăm este Sâmbătă.

Folosind **Random Forest Regression**, am obținut rezultate similare, dar cu mici diferențe în scoruri:

```
Feature: faces, importance: 0.03229945552729712
Feature: smiles, importance: 0.030505650451851267
Feature: (17, 22), importance: 0.02782300581823638
Feature: (12, 17), importance: 0.02261118534530509
Feature: Thursday, importance: 0.021317839784769088
Feature: Friday, importance: 0.019894899423591274
Feature: Tuesday, importance: 0.01980649762469207
Feature: Saturday, importance: 0.01961744254251792
Feature: Wednesday, importance: 0.01915201965871097
Feature: Monday, importance: 0.019023200470695285
Feature: (8, 12), importance: 0.017949544289794554
Feature: (2, 8), importance: 0.016806902898339
Feature: Sunday, importance: 0.016323820021096447
```

```
Feature: word: love, importance: 0.015802036719741546
Feature: word: thi, importance: 0.014947546514970973
Feature: word: thank, importance: 0.009251241901395282
Feature: word: time, importance: 0.00868640560048905
Feature: word: ucl, importance: 0.008504542798358081
Feature: word: para, importance: 0.007594194714588147
Feature: word: video, importance: 0.0074664378044882315
Feature: word: like, importance: 0.00746363798407831
Feature: word: look, importance: 0.0068989256205657275
Feature: word: todo, importance: 0.0067602591862999635
```

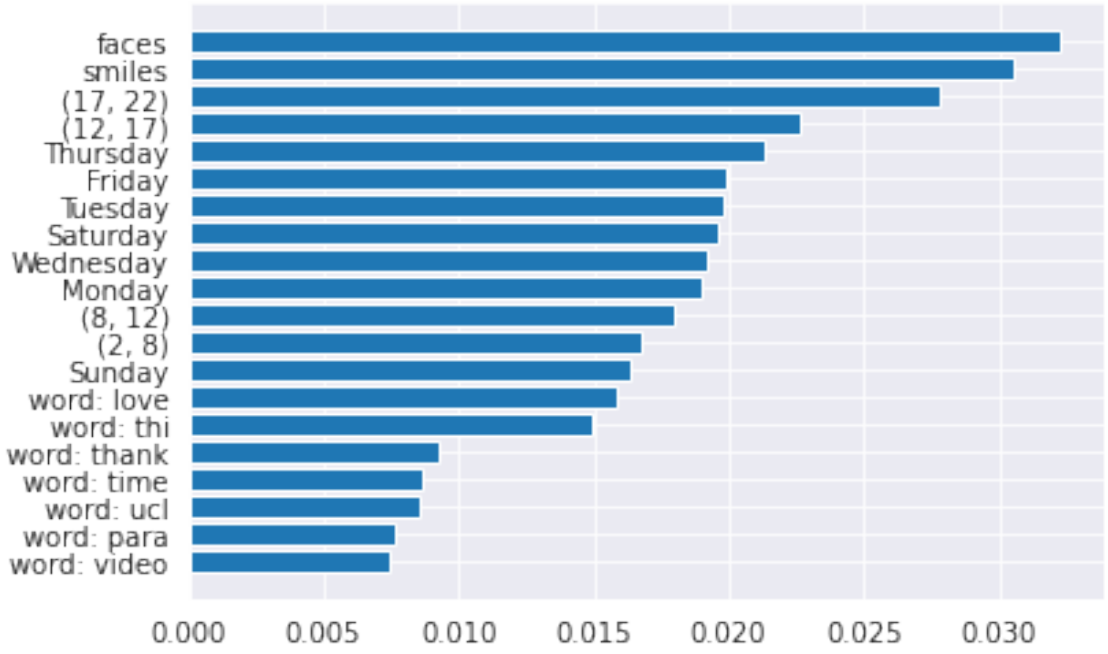

Figura 2.9: Importanța caracteristicilor - Random Forest

Spre deosebire de ultimul model, putem observa următoarele diferențe

- Ziua cea mai bună în care să postăm este Joi
- Orele cele mai bune de postat sunt între 17:00 și 22:00, urmate de orele 2:00-8:00.

2.4 Impărțirea dataset-ului

Am împărțit datasetul nostru în 10% test, 10% validation și 80% train. Pentru fiecare epocă în parte am realizat statistici atât pe setul de antrenare cât și pe cel de validare.

De asemenea, am impărțit dataset-urile de test, train si validation in batch-uri de 64, randomizate pentru train data. Aceste batch-uri le-am construit folosind clasa DataLoader din torch.utils.data.

```
# Split dataset in train, test and validation
dataset = pd.read_csv(self.CURRENT_FEATURES_FILE)
dataset = dataset.fillna(0)
msk = pd.np.random.rand(len(dataset)) < 0.8
self.train_ft = dataset[msk]
dataset2 = dataset[~msk]
msk2 = pd.np.random.rand(len(dataset2)) < 0.5
self.validation_ft = dataset2[~msk2]
self.test_ft = dataset2[msk2]

# Split datasets in batches
self.test_dl: DataLoader = DataLoader(Dataset(self.test_ft), batch_size=64,
shuffle=False)
self.validation_dl: DataLoader = DataLoader(Dataset(self.validation_ft),
batch_size=64, shuffle=False)
self.train_dl: DataLoader = DataLoader(Dataset(self.train_ft), batch_size=64,
shuffle=True)
```

2.5 Definirea modelului

Vom folosi un model Secvențial cu 5 layere de $215 * 1028, 1028 * 256, 256 * 128, 128 * 32$ și $32 * 1$. Între fiecare layer am adăugat câte o funcție de activare ReLU, iar la final am adăugat un dropout de 0.3 pentru a evita overfitting-ul. Desigur, înainte de a adăuga datele în model le-am aplicat o funcție de normalizare.

Modelul nostru l-am realizat folosind librăria torch din Python, mai precis, am moștenit clasa torch.nn.Module, adăugând un clasificator Secvențial cu layerele descrise anterior.

```
class Model(torch.nn.Module):
    def __init__(self, p: float = .3):
        super().__init__()
        self.layers = torch.nn.Sequential(
            torch.nn.Linear(215, 1028),
            torch.nn.ReLU(),
            torch.nn.Linear(1028, 256),
            torch.nn.ReLU(),
            torch.nn.Linear(256, 128),
            torch.nn.ReLU(),
            torch.nn.Linear(128, 32),
            # Dropout layer
            torch.nn.Dropout(p),
            torch.nn.ReLU(),
            torch.nn.Linear(32, 1)
        )


    def forward(self, x):
        # Normalize input
        x = normalize(x, p=2, dim=0)
        return self.layers(x)
```

Pentru antrenarea acestui model vom folosi funcția de optimizare Adam, cu learning rate = 1e-4 = 0.0001. Această funcție de optimizare se utilizează pentru optimizările gradiente de primul ordin a funcțiilor stochastice obiective, bazate pe estimările momentelor de ordin mai mic. [13]

```
self.optimizer = torch.optim.Adam(self.model.parameters(), lr=1e-4)
```

Ca și funcție de pierdere, vom folosi L1 Loss (Least Absolute Deviation), ce are ca scop minimizarea erorii reprezentate de suma tuturor diferențelor absolute dintre valorile prezise și cele reale. [17]

$L1Loss = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$

```
self.loss_fn = torch.nn.L1Loss()
```

2.5.1 Evaluarea modelului

Pentru a putea evalua modelul nostru vom folosi următoarele metrici:

- MSE - Mean Square Error este calculat drept media pătratelor diferențelor dintre valorile prezise și cele adevărate. $MSE = \frac{1}{n} * \sum_1^n y_{true} - y_{predicted}$. [12]
- MAE - Mean Absolute Error este calculat drept media erorilor absolute. MAE crește în mod liniar în funcție de câte erori întâmpină. $MAE = \frac{1}{n} * \sum_1^n abs(y_{true} - y_{predicted})$. [12]
- Scorul R^2 reprezintă acuratețea aproximată cu care prezice modelul. Deși valorile acestei metrici sunt între 0 și 1, aceasta poate lua și valori negative.

2.6 Antrenarea modelului

2.6.1 Partea I

Pentru a antrena modelul, am păstrat 214 de feature-uri, printre care 200 de caracteristici extrase din descrieri, 2 din imagini și restul din dată și ora la care a fost adăugată postarea. Pentru fiecare postare am folosit ca metrică raportul dintre like-uri și urmăritori. Acest raport poate fi înmulțit ulterior cu numărul de urmăritori pentru a obține un număr de like-uri concret pentru fiecare postare în parte.

Pentru acest model am obținut următoarele statistici:

- După 3 epochi:

```
1    MSE: 0.0012158110589540118
2    Mean absolute error: 0.017973597316153472
3    R2 score: -99.97921983832735
```

- După 1000 epochi:

```
1    MSE: 0.0011458757148844695
2    Mean absolute error: 0.017063193946872352
3    R2 score: -16.014154436814696
```

Pentru o analiză mai exactă, am înmulțit rezultatele primite (ce reprezintă raportul dintre like-uri și followers) cu numărul de urmăritori pentru fiecare cont în parte. În final am realizat următorul grafic ce afișează diferențele dintre like-urile prezise și like-urile primite. M1 - 3 epoci = diferențele după 3 epoci, M2 - 1000 epoci = diferențele după 1000 de epoci.

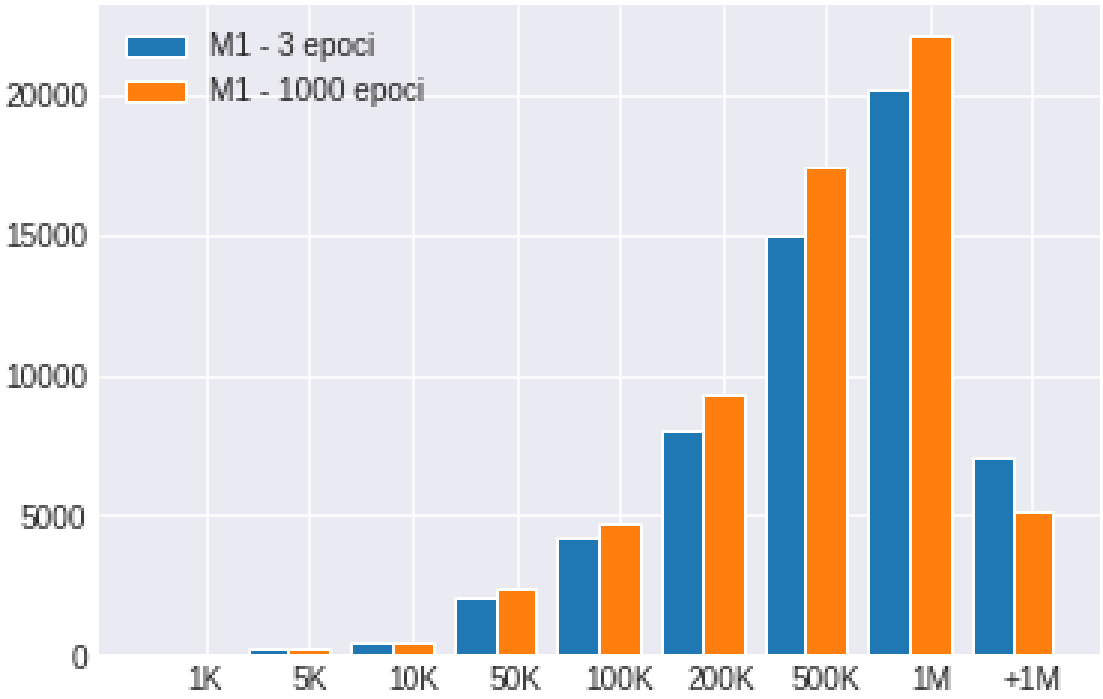

Figura 2.10: Modelul 1 - 3 vs 1000 de epoci

Din acest grafic putem observa că cele mai mari diferențe de like-uri sunt de mai puțin de 1M, rezultate destul de rele ținând cont că utilizatorii de la care am extras postările au în medie 867K de like-uri, iar numărul maxim de like-uri pentru o postare este de 19M.

2.6.2 Partea II

Pentru a îmbunătății modelul nostru, vom folosi ca metrică raportul dintre like-uri și media aprecierilor raportate la contul postării. Dezavantajul acestei metode este reprezentat de necesitatea unei informații în plus pentru fiecare postare (ce urmează a fi prezisă), mai exact media like-urilor postărilor trecute. Prin această abordare, utilizatorul va fi nevoit să aiba cel puțin 1 postare pe cont.

Astfel, dupa rularea a 500 de epoci folosind noua metrică, am obținut următoarele rezultate:

```
MSE:  0.776016050386604
Mean absolute error:  0.5172142815730558
R2 score:  -2.0881367596652693
```

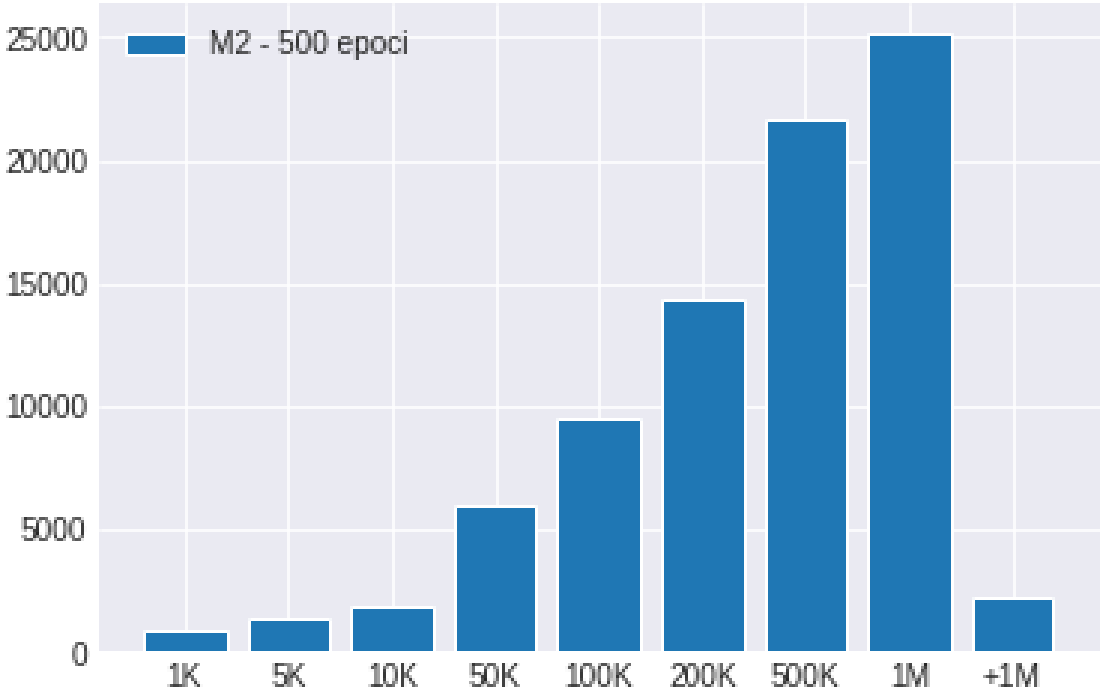

Figura 2.11: Modelul 2

Din graficul precedent putem spune că avem rezultate destul de satisfăcătoare, ce pot fii imbunătățite prin rularea a cât mai multor epoci de antrenare. De asemenea, putem vedea comparația dintre modelul curent și ultimul model (ce folosea ca metrică numărul like-uri pe followeri).

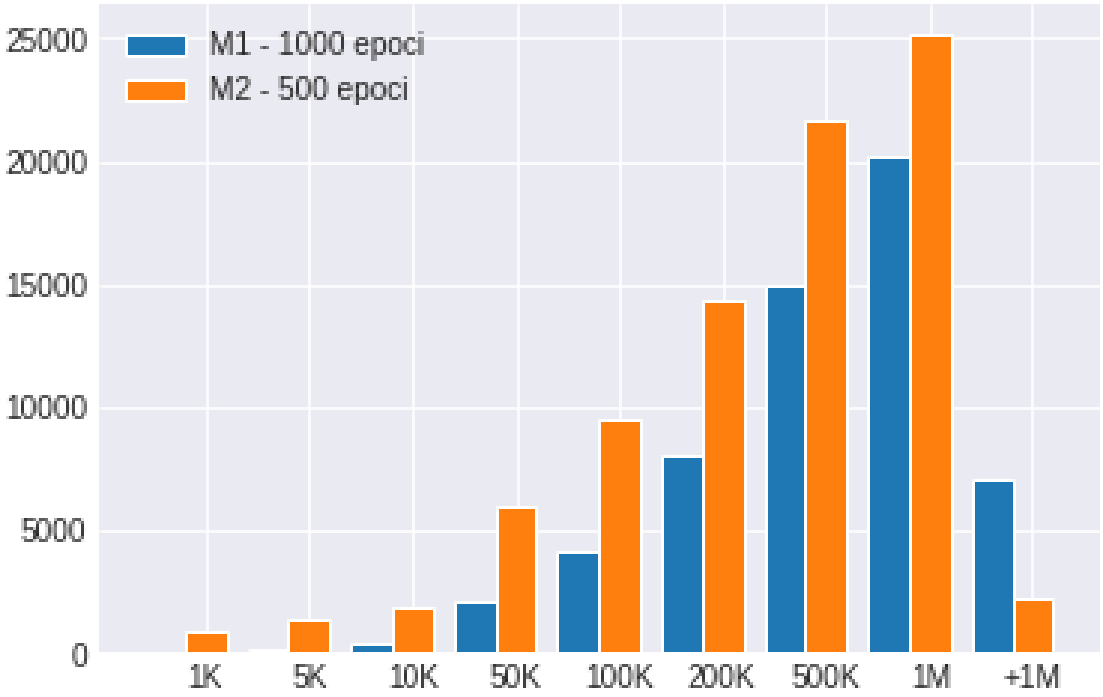

Figura 2.12: Comparație modele 1/2

Prin folosirea mediei aprecierilor pe like-uri primite ca și metrică ne vom putea raporta la fiecare cont de instagram, ținând cont de audiența și media like-urilor fiecărui cont în parte.

Spre deosebire de urmăritori, istoricul like-urilor unui cont arată ce tip de audiență are și câți dintre urmăritori săi sunt dispuși să interacționeze cu postările sale.

Antrenând modelul pe raportul de like-uri pe media lor, vom obține o generalizare mai amplă asupra influenței caracteristicilor extrase. Ca de exemplu din acest model putem observa că pentru fiecare persoană adăugată în poză vom obține 0.1 like-uri raportate la media lor. Ca și contra exemplu, dacă am utiliza raportul umăritorilor, nu am obține un rezultat prea concret, pentru că nu știm câți dintre acei urmăritori vor aprecia postarea.

Mai jos putem observa acuratețea modelului nostru măsurată în diferențele de like-uri prezise și actuale.

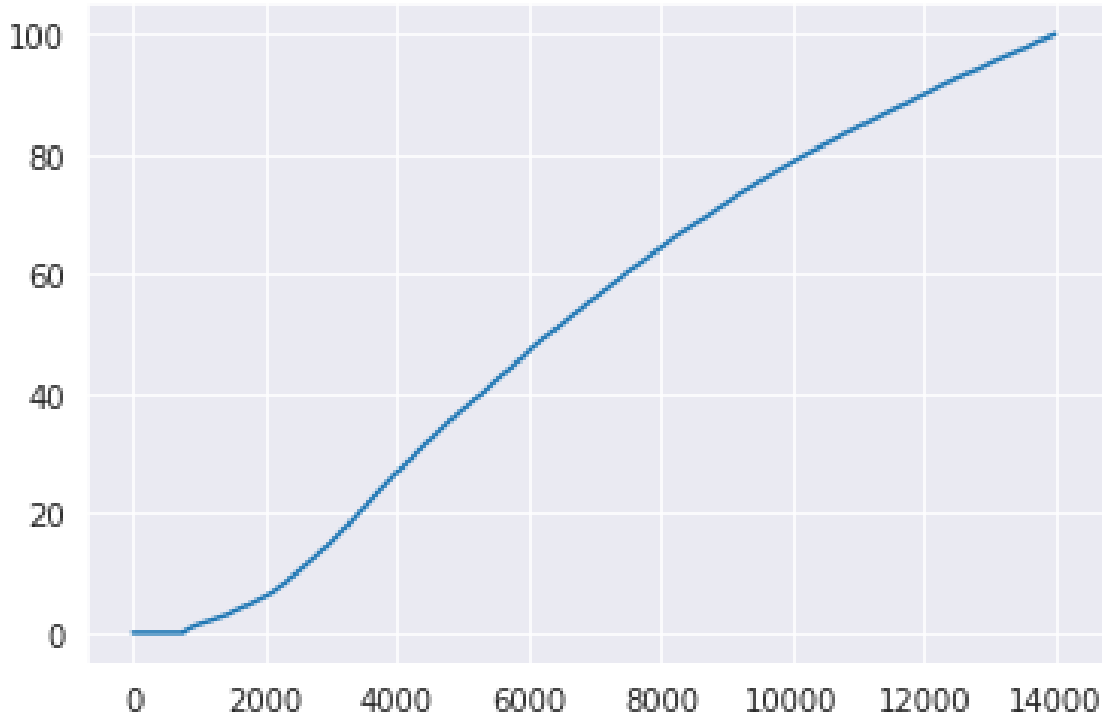

Figura 2.13: Acuratețea modelului

Capitolul 3

Aplicația Mobilă

3.1 Technologii folosite

Pentru a realiza aplicația mobilă vom folosi următoarele technologii:

- Flask pentru Backend
- MySql pentru baza de date
- Flutter pentru Frontend - Aplicația mobilă

3.1.1 Backend - Flask

Pe partea de backend vom utiliza Flask [6], un micro-framework web scris în Python [3]. Spre deosebire de Django sau alte framework-uri populare de backend, Flask este ușor de inițializat, fără bază de date incorporată, validări de formulare sau structuri predefinite de inițialiare a modulelor. Fiind un framework light, este recomandat pentru proiectele mici.

Pentru a realiza o conexiune la baza de date, vom folosi librăria PyMySql, librărie bazată pe PEP249, ce suportă conexiunea la baza de date Mysql. [7]

Aplicația o vom hosta la adresa: 176.126.237.70:3002, unde o să ruleze sub formă de serviciu în mediul development, utilizând versiunea 3.10 de Python și environmentul virtual (virtualenv) pentru gestionarea pachetelor.

3.1.2 Frontend - Flutter

Aplicația mobilă o vom realiza folosind Framework-ul creat de Google, Flutter [4], ce este scris in limbajul Dart [8]. Fiind un framework Multi-Platform, vom putea realiza 2 aplicații, respectiv una pentru Android și una pentru iOS, folosind același cod.

3.2 Baza de date

Am folosit o baza de date de tip MySql pentru a salva următoarele:

- Conturile utilizatorilor in tabela Users, alături de metoda de autentificare 3rd party sau parola hash-uită.
- Sesiunile de autentificare (Sessions), tabelă folosită și pentru a salva parolele înainte de resetarea lor.
- Conturile de instagram și statisticile lor le-am salvat în tabela InstagramAccounts. Această tabela este legată de conturile utilizatorilor și menține statisticile legate de conturile de instagram a utilizatorilor (număr de urmăritori, media like-urilor, username, etc.). Folosim această tabelă pentru a nu supra-solicita api-ul instagram ce ne limitează la un număr de request-uri pe zi.
- Sesiunile de autentificare pentru conturile private de instagram le stocăm în tabela InstagramTokens. Utilizatorii nu au access la aceasta tabelă, dar sesiunile de autentificare din ea sunt folosite pentru a extrage date despre conturile utilizatorilor.

Mai jos putem observa diagrama aferentă bazei noastra de date, descrisă anterior:

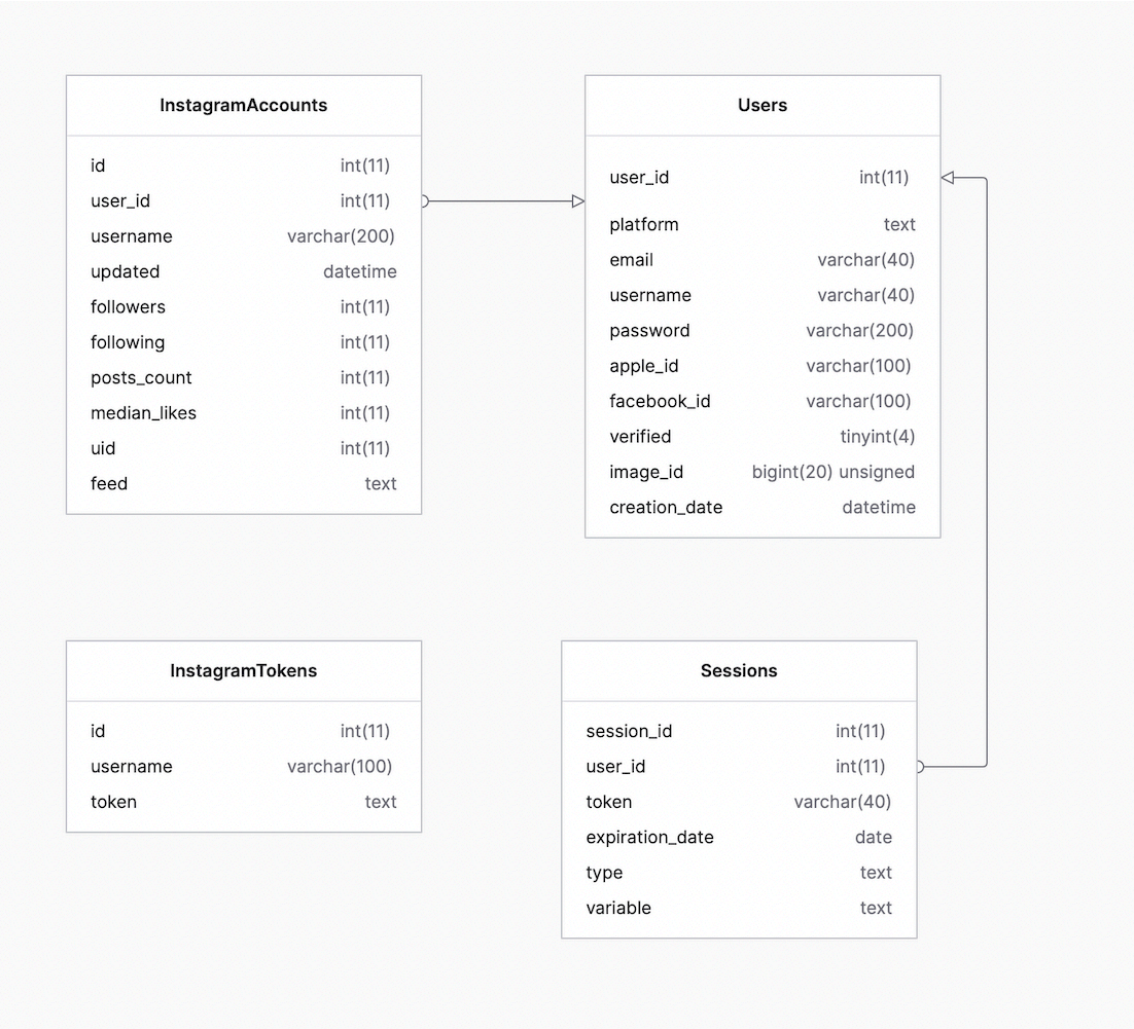

Figura 3.1: Diagrama bazei de date

3.3 Autentificarea

Pentru a reține informații utile legate de conturile de instagram a utilizatorilor, vom integra o funcție de autentificare, folosind diverse metode:

- Autentificarea cu Apple folosind librăria sign_in_with_apple pentru frontend și api-ul oficial Apple: appleid.apple.com pentru validarea conturilor pe backend. [10]
- Autentificarea cu Facebook folosind librăria flutter_facebook_auth pentru frontend și api-ul oficial Facebook: graph.facebook.com pentru backend. [18]
- Autetificarea prin Email, ce include crearea contului și recuperarea parolei prin trimiterea unui link de resetare.

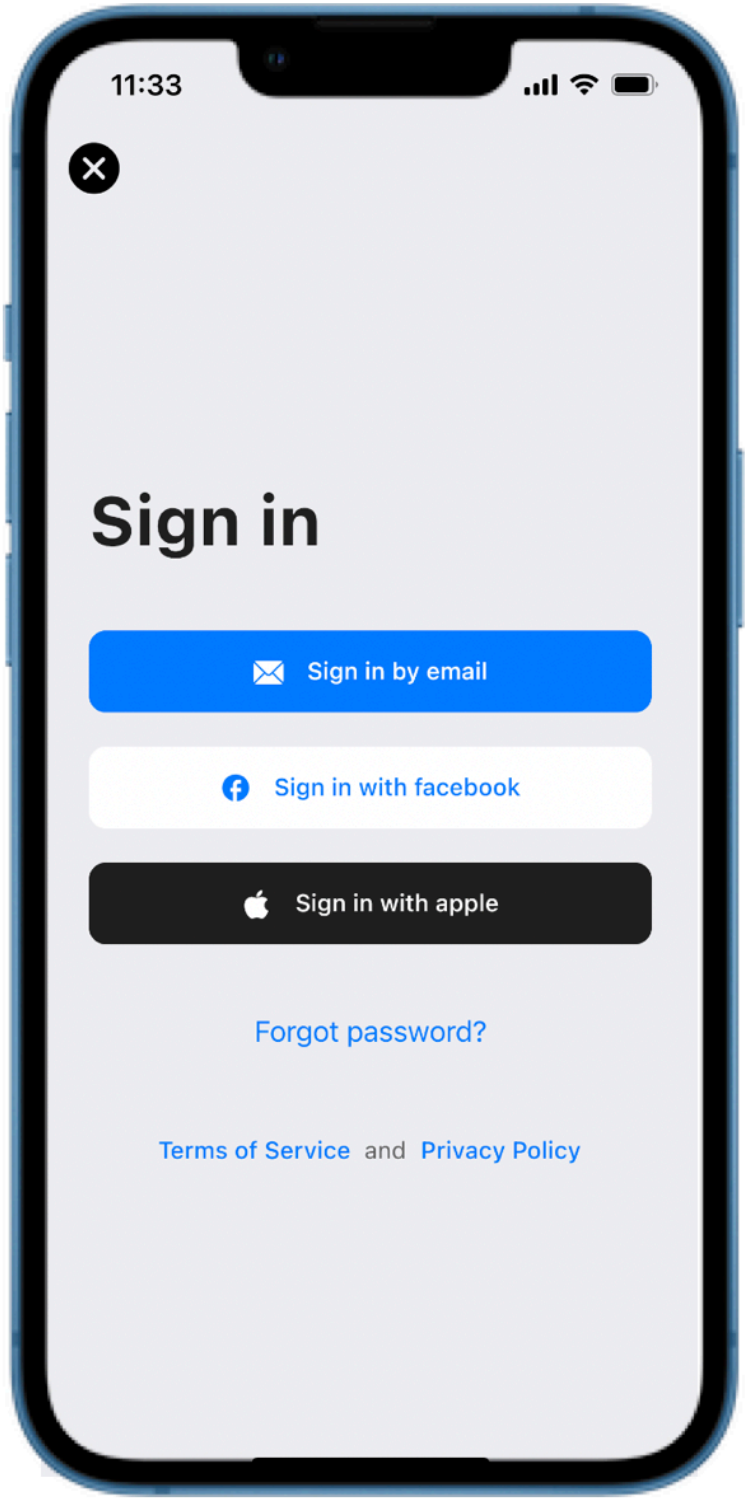

Figura 3.2: Ecranul de autentificare

3.4 Relația cu contul de instagram al utilizatorului

Pentru a prelua cele 2 metrici necesare în predicția numărului de like-uri, respectiv media like-urilor precedente și numărul de urmăritori, vom conecta contul de instagram

al utilizatorului la contul său din aplicație.

Desigur, pentru a avea access la statisticile contului, acesta trebuie să fie public și să conțină cel puțin 1 postare. O altă abordare ar fi să cerem utilizatorilor să-și introducă pe lângă username, parola. În schimb prin preluarea informațiilor publice putem oferi o securitate mai mare datelor extrase de la utilizator.

Pe partea de backend, extragem datele utilizatorului folosind contul de instagram deja asociat aplicației (la care user-ul nu are access) pentru a accesa api-ul oficial instagram. [1]

```
def getAccountDetails(self, user_id, username):
    username_info = self.api.username_info(username)['user']
    user_feed = self.api.username_feed(user_name=username, count=64)['items']

    all_likes = [x['likes']['count'] for x in user_feed]
    median_likes = sum(all_likes) / len(all_likes)
    self.insert("InstagramAccounts", {
        "user_id": user_id,
        "username": username,
        "followers": username_info['follower_count'],
        "following": username_info['following_count'],
        "posts_count": username_info['counts']['media'],
        "median_likes": round(median_likes),
        "uid": username_info['id'],
        "feed": json.dumps(user_feed)
    }, check_table_column=False)

    return {
        "username": username,
        "followers": username_info['follower_count'],
        "following": username_info['following_count'],
        "posts_count": username_info['counts']['media'],
        "median_likes": round(median_likes),
        "uid": username_info['id']
    }
```

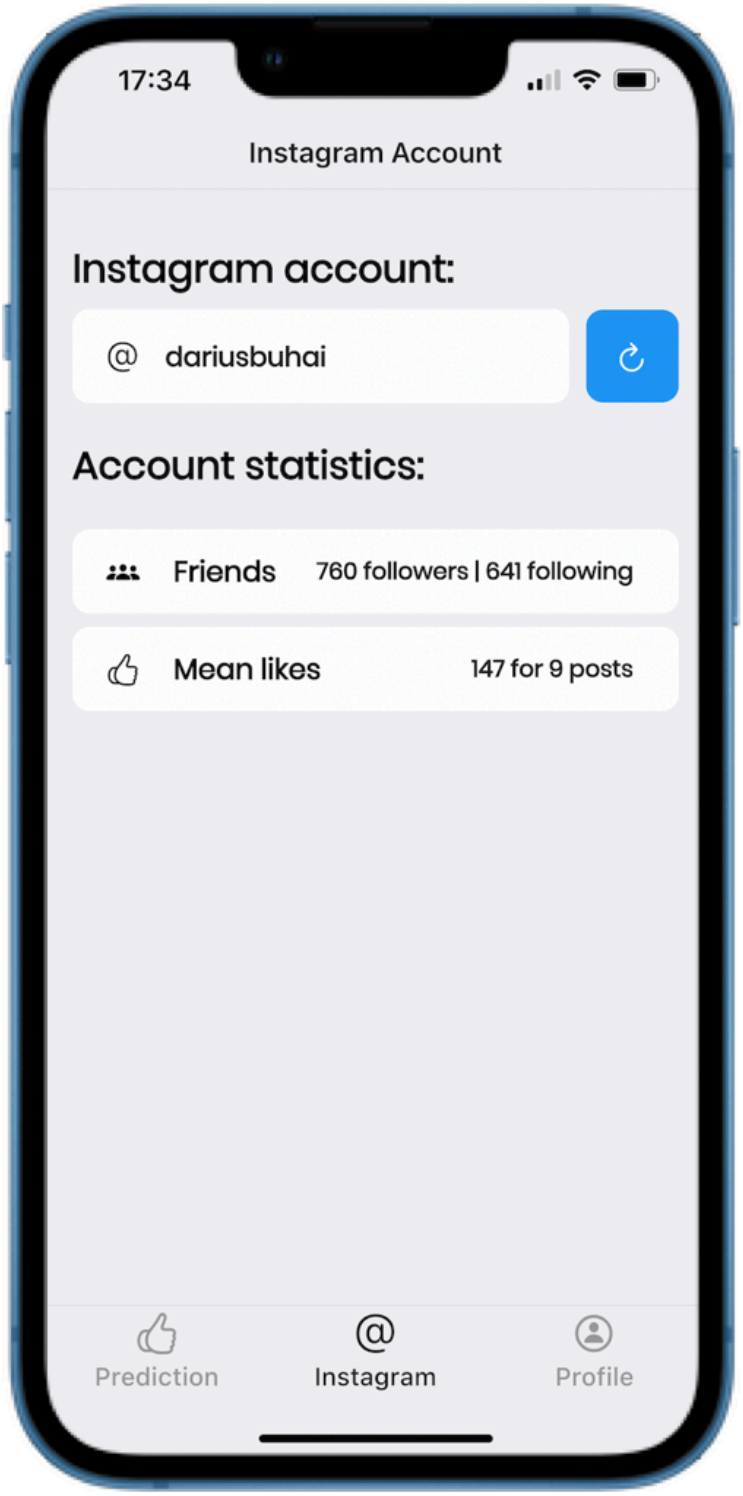

Figura 3.3: Statisticile contului de instagram

3.5 Predicția postărilor

Pentru a putea apela modelul nostru și a oferi o predicție asupra numărului de like-uri pentru o postare, utilizatorul are de completat următoarele date:

- Poza pe care dorește să o posteze. Din această sunt extrase și afișate numărul de fețe și zâmbete.
- Descrierea postării
- Dată și ora la care dorește să o posteze
- În cazul în care nu și-a conectat contul de instagram, acesta mai trebuie să completeze media like-urilor și numărul de urmăritori.

Procesul de predicție este împărțit în următorii pași:

1. Utilizatorul încarcă imaginea, această este trimisă la api-ul nostru ce aplică modelul OpenCV descris în capitolul 2.3.1. În schimb primește imaginea formatată cu persoanele evidențiate, numărul de persoane și de zâmbete.
2. Sunt completate datele referitoare la postare (descriere, dată și ora). Din aceste date vor fii extrase caracteristicile necesare predicției (ziua din săptămână, token-urile utilizate în descriere etc.).
3. Se preiau automat detalii referitoare la contul de instagram al utilizatorului sau se completează manual.
4. După ce utilizatorul cere o predicție, se aplică pașii de preprocesare (descris în capitolul 2.2), feature_extraction (descris în 2.3.2 și 2.3.3) , iar în final este aplicat cel mai bun model salvat.
5. Rezultatele sunt înmulțite cu media like-urilor și afișate pe ecran.

```
@staticmethod
def predictPostLikes(date_time: datetime, description: str,
    followers: int = 0, faces: int = 0, smiles: int = 0, mean_likes=None):
    post = {
        "smiles": smiles,
        "faces": faces,
        "followers": followers,
        "description": description,
        "taken_at": datetime.timestamp(date_time),
    }
    posts = Preprocessing.preprocessPosts(pd.DataFrame([post]))
    features_extraction = FeaturesExtraction(posts)
    posts = features_extraction.extractPostsFeatures()
    regressor = Regressor(prediction_mode=True)
    regressor.loadModel()
    response = regressor.predict(posts.iloc[0])
    if mean_likes is not None:
```

```
        return round(float(response) * float(mean_likes)),
        round(float(response), 3)
    return None, round(float(response), 3)
```

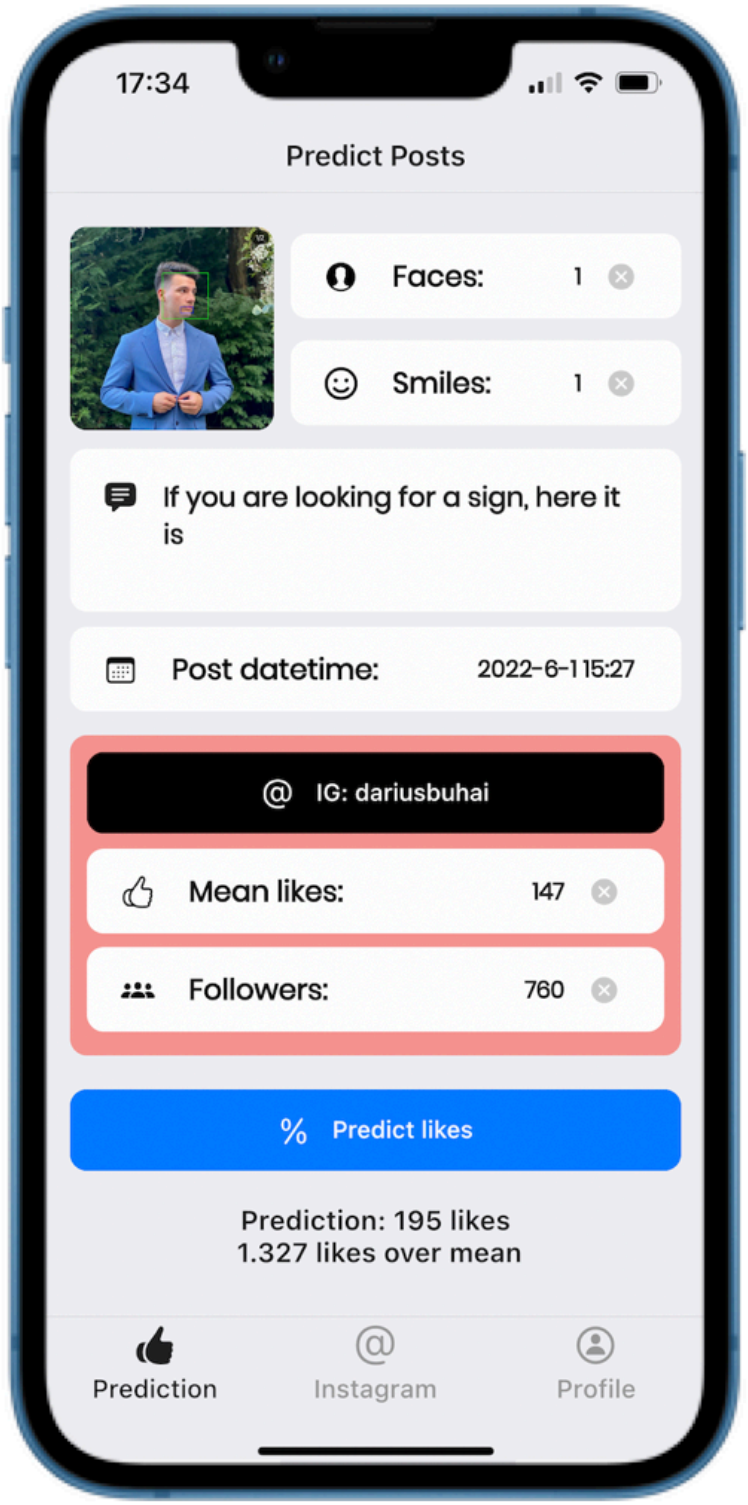

Figura 3.4: Vizualizarea predicției

Capitolul 4

Concluzie

4.1 Exemple de predicții

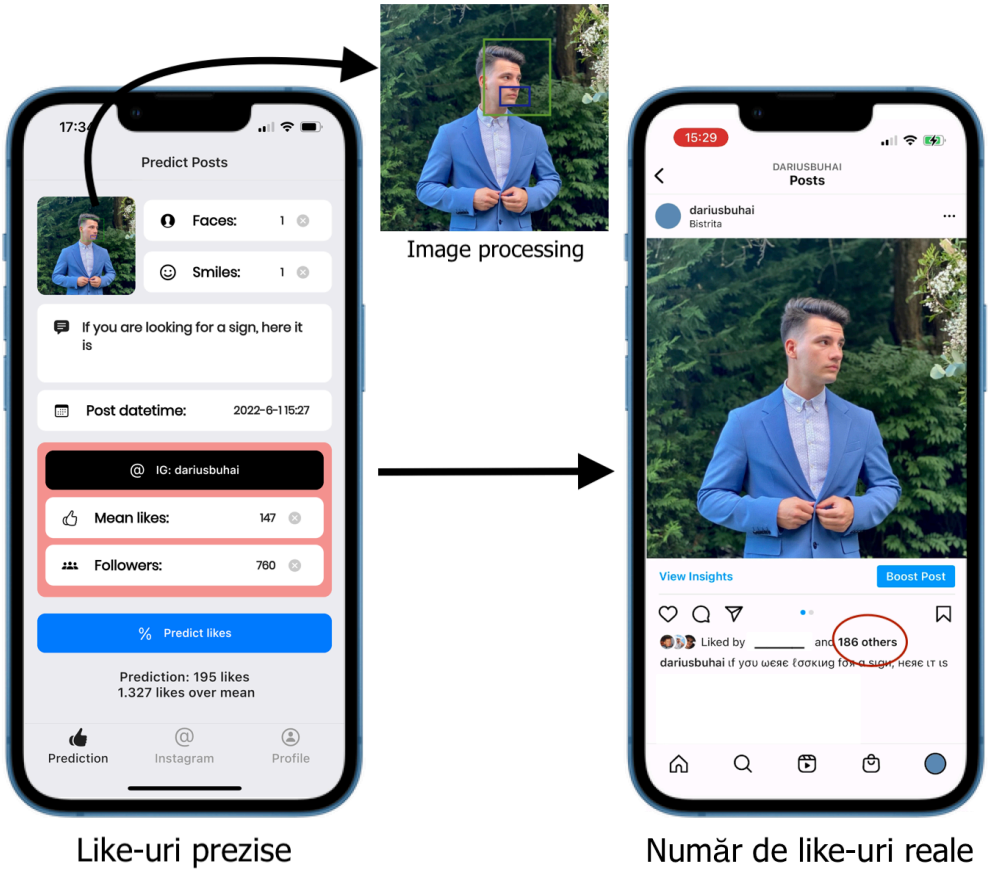

Figura 4.1: Exemplu predicție

În exemplul de mai sus am încărcat contul meu de instagram, preluând numărul de urmăritori și media like-urilor. De asemenea am încărcat imaginea postată ce conține o singură persoană detectată de modelul OpenCV, descrierea, data și ora la care a fost

postată.

Algoritmul nostru de predicție a estimat un număr de 195 de like-uri pentru postarea respectivă, mai precis, 1.327 aprecieri / medie, în realitate am obținut 186 de like-uri, ceea ce nu e prea departe de predicția noastră.

Pe lângă acest exemplu vizual, putem vedea cum performează algoritmul nostru pe setul nostru de testare. Astfel pentru 10 postări selectate random, am obținut următoarele predicții:

```
1. Actual likes: 7,886,085; Predicted: 6,919,726
2. Actual likes: 9,332,713; Predicted: 10,841,930
3. Actual likes: 10,255,392; Predicted: 24,439,176
4. Actual likes: 7,873,294; Predicted: 14,254,058
5. Actual likes: 6,627,776; Predicted: 6,299,707
6. Actual likes: 10,752,159; Predicted: 28,721,793
7. Actual likes: 10,254,770; Predicted: 14,114,124
8. Actual likes: 7,910,983; Predicted: 10,809,952
9. Actual likes: 2,777,085; Predicted: 621,159
10. Actual likes: 4,802,595; Predicted: 2,312,874
```

Cu toate că predicțiile nu sunt întotdeauna reale pentru conturile mici (sub 10.000 de urmăritori), pentru conturile mari de instagram predicțiile încep să devină cât mai realiste având o marjă de eroare din ce în ce mai mică pe măsură ce numărul de urmăritori crește.

4.2 Îmbunătățiri și Probleme întâmpinate

- În primul rând, pentru a obține predicții mai reale, ar trebui să adăugăm mai multe caracteristici de antrenare a modelului. Pe lângă descrierea postării și data la care a fost postata, ar trebui să acordăm o importanță mult mai ridicată imaginilor, deoarece influențează în cea mai mare proporție postările de instagram. Astfel, am putea lua 2 abordări:
  - Cea mai ușoară abordare o reprezintă antrenarea unui model separat folosind Image Processing și algoritmi de Computer Vision. Alături de modelul acesta, am putea crea un alt model ce să includă data, ora și descrierea postării. În final ar trebui să găsim o metodă de combinare a acestor două modele, fie prin media rezultatelor sau prin combinarea inputului într-un singur model.
  - O altă abordare mai complexă ar fi să extragem cât mai multe trăsături din imagini folosind în continuare OpenCV și modele preantrenate. Astfel, pe lângă numărul de fețe și zâmbete din imagini, am putea analiza: calitatea pozei, tipul de background, diverse trăsături ale fețelor, alte obiecte sau ființe prezente în poze (animale de companie, plante, produse etc.). Problema acestui

tip de antrenare o reprezintă complexitatea extragerii diferitelor trăsături din imagini, cât și alegerea lor în mod eficient.

- În al doilea rând, pentru o predicție mai eficientă vom avea nevoie de un set de date mult mai mare (având în prezent >270.000 de postări) dar și mai diversificat. Astfel, pe lângă mărirea setului de date până la >1000 de influenceri și >1.000.000 de postări, ar trebui să adăugăm și influenceri cu un raport mic de like-uri pe număr de urmăritori pentru a înțelege și ce caracteristici fac o postare să eșueze. În același mod, în crearea vocabularului ar trebui să includem și cuvinte negative ce impactează în mod negativ o postare (hate speech sau opinii controversate expuse în descrieri).
- În ultimul rând, antrenarea modelului trebuie să fie specifică pentru fiecare cont în parte și actualizat constant. Spre exemplu, o postare publicată în 2019 în aer liber la o terasă nu ar obține aceeași audiență negativă ca și o postare publicată în contextul pandemiei (2021) în același loc, prin nerespectarea distanțării sociale.

În concluzie, prezicerea numărului de like-uri a unei postări de instagram este și va rămâne în continuare o temă greu de rezolvat datorită multitudinii de factori implicați în prezicerea preferințelor publicului. Prin această lucrare am încercat să prezint date, metrici, statistici și câteva moduri de prezicere a like-urilor pe instagram.

Bibliografie

[1] URL: https://developers.facebook.com/docs/instagram-api/.

[2] URL: https://github.com/ping/instagram_private_api.

[3] URL: https://www.python.org/doc/.

[4] URL: https://flutter.dev/.

[5] URL: https://docs.opencv.org/4.x/d7/dbd/group__imgproc.html.

[6] URL: https://flask.palletsprojects.com/en/2.1.x/.

[7] URL: https://pypi.org/project/PyMySQL/.

[8] URL: https://dart.dev/.

[9] „Bag of Words", în (Mai 2022), URL: https://en.wikipedia.org/wiki/Bag-of-words_model.

[10] Aamish Baloch, „Sign in with Apple", în (2019), URL: https://gist.github.com/aamishbaloch/2f0e5d94055e1c29c0585d2f79a8634e.

[11] Jason Brownlee, „Calculate feature importance", în (Mar. 2020), URL: https://machinelearningmastery.com/calculate-feature-importance-with-python/.

[12] Jason Brownlee, „Regression Metrics for Machine Learning", în (Ian. 2021), URL: https://machinelearningmastery.com/regression-metrics-for-machine-learning/.

[13] Jimmy Ba Diederik P. Kingma, „Adam: A Method for Stochastic Optimization", în (2017), URL: https://arxiv.org/abs/1412.6980.

[14] Corentin Dugué, „Predict number of likes on instagram", în (Mai 2017), URL: https://towardsdatascience.com/predict-the-number-of-likes-on-instagram-a7ec5c020203.

[15] Vairaprakash Gurusamy, „Preprocessing Techniques for Text Mining", în (Oct. 2014), URL: https://www.researchgate.net/profile/Vairaprakash-Gurusamy/publication/273127322_Preprocessing_Techniques_for_Text_Mining/links/54f8319e0cf210398e949292/Preprocessing-Techniques-for-Text-Mining.pdf.

[16] Prasert Kanawattanachai, „Top 1000 Instagram Influencers Dataset”, în (Feb. 2022), URL: https://www.kaggle.com/datasets/prasertk/top-1000-instagram-influencers.

[17] Amit Shekhar, „What Are L1 and L2 Loss Functions?”, în (Aug. 2019), URL: https://afteracademy.com/blog/what-are-l1-and-l2-loss-functions.

[18] Prerna Srivastava, „Sign in with Facebook”, în (Iul. 2020), URL: https://www.geeksforgeeks.org/facebook-login-using-python/.

[19] Guilherme Regos Zamorano, „Predicting the Popularity of Instagram Posts”, în (Mai 2019), URL: https://towardsdatascience.com/predicting-the-popularity-of-instagram-posts-deeb7dc27a8f.