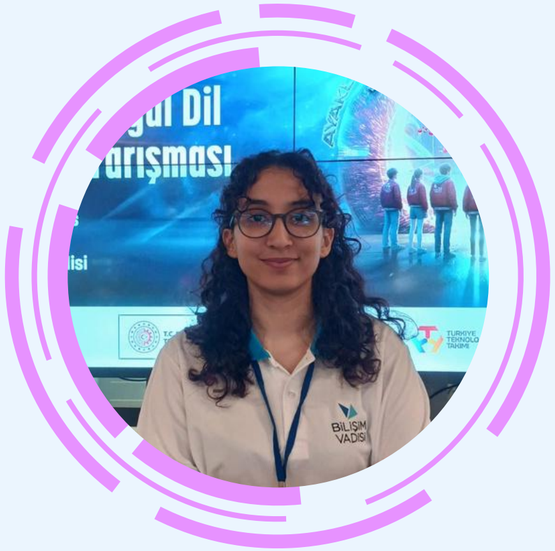
Fundamentals of Machine Learning

REVIEW AND JOB MATCHING

Meet Our TEAM



www.review_and_job_matching .com

Problem

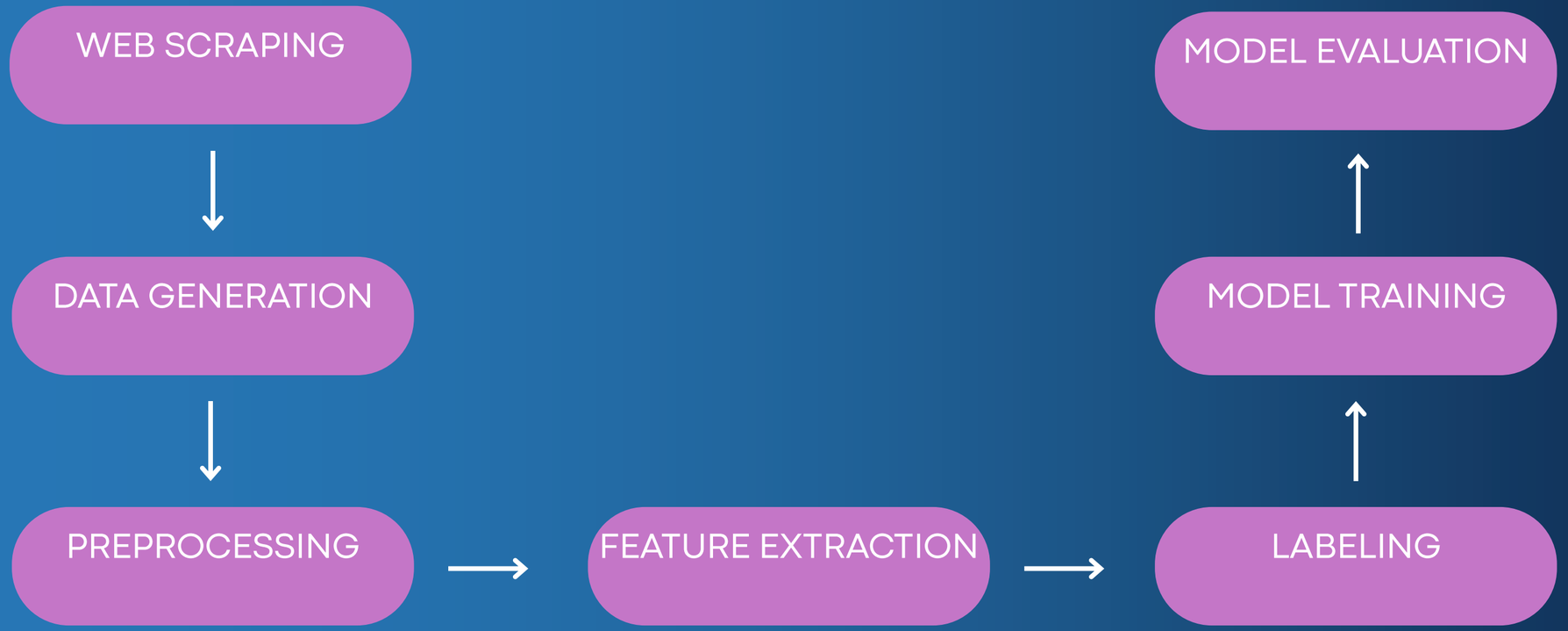
The job search process is a painful experience that everyone has gone through at least once. During this process, we are forced to go through thousands of job listings to find the job that matches our criteria and qualifications.

www.review_and_job_matching .com

Solution

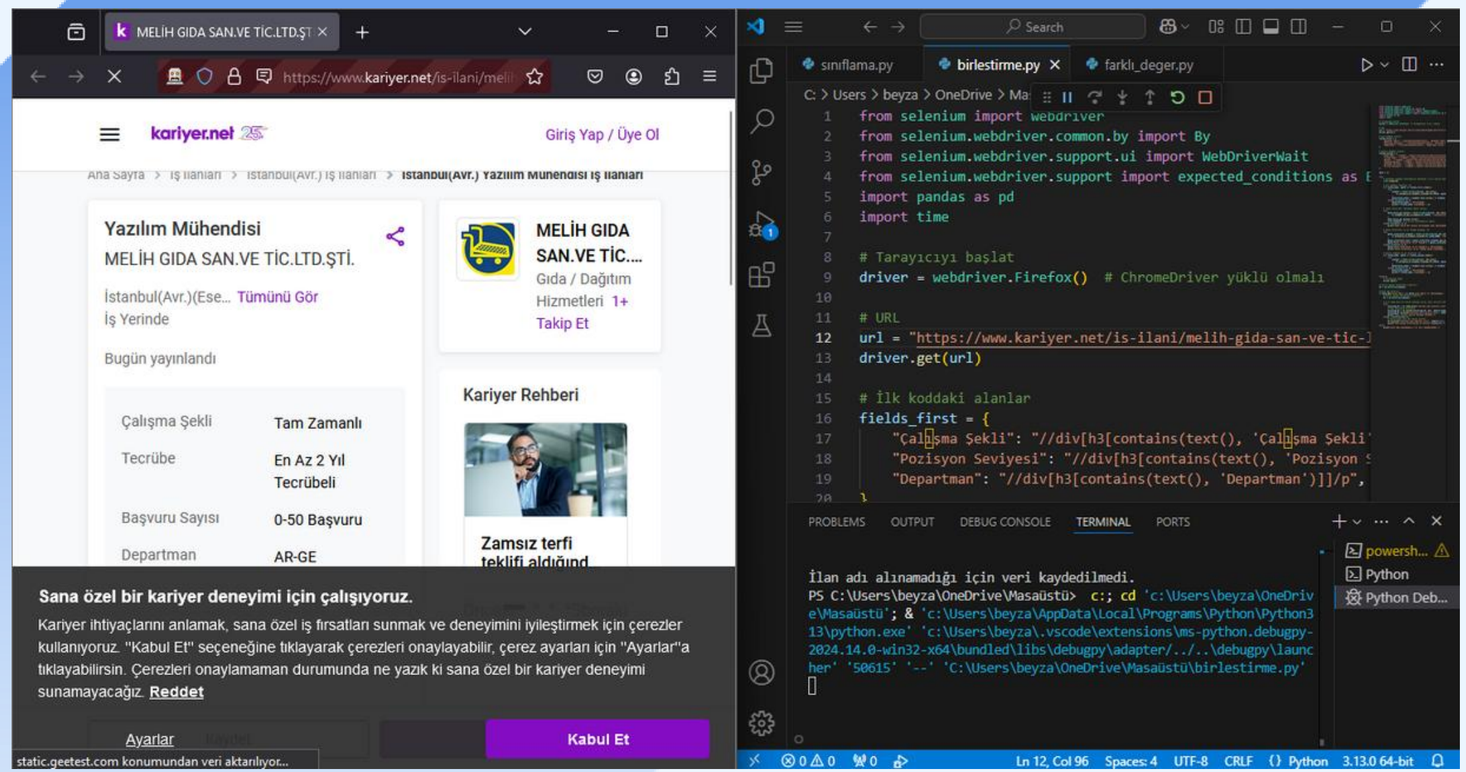
This project aims to enhance resume evaluation and job-matching processes. Fundamentally, it compares candidates' resumes with suitable jobs and reflects a compatibility score between the candidate and the job. By leveraging various technologies, the project seeks to automate the resume review and job-matching process, making these operations faster and more efficient.

Project Steps



DATASET CREATION

We created two main datasets for our project: one containing resume information and the other detailing job postings. For the resume dataset, we generated synthetic data using Gemini and found suitable pre-existing datasets on Kaggle. We processed the Kaggle data by organizing it into columns and extracting necessary features through AI prompts. While the Kaggle data contained sufficient detail, the data generated by Gemini was inadequate, leading to the decision not to use it. To build the job posting dataset, we collected data from platforms like LinkedIn and Kariyer.net using web scraping methods with the Selenium library.



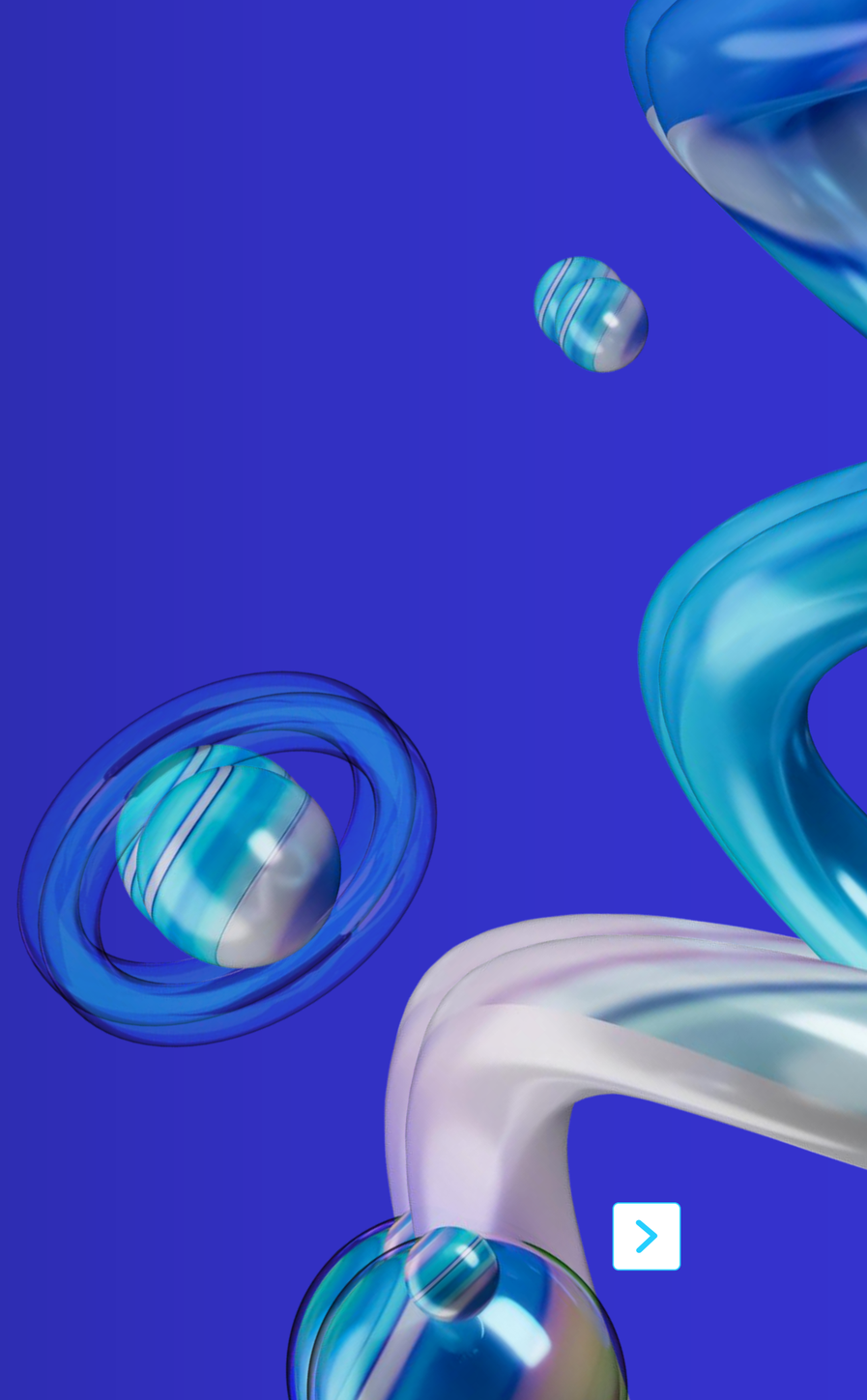
PREPROCESSING

1. Removing Whitespaces:
   - Leading and trailing whitespaces are removed from the data.
   - Unnecessary spaces are cleaned to standardize the data.
2. Converting to Lowercase:
   - Data is converted to lowercase to ensure consistency.
3. Removing Parentheses:
   - Explanations or information within parentheses are removed from the text.
4. Using a Word Dictionary:
   - Similar or synonymous terms in fields like department, organization, or language are merged and standardized.
5. Splitting and Processing by Comma:
   - Data is split by commas, and each part is processed separately.
6. Date Formatting:
   - Various date formats are standardized to extract only the year.
   - Phrases like "Present" or "Still Continuing" are processed to assign relevant years.
7. Feature Extraction:
   - Specific features (e.g., graduation degree) are extracted from the text.
8. Handling Missing Data:
   - Default values (e.g., "Not Specified") are assigned for missing or undefined values.

Merging of Datasets

Each CV line was written in multiple combinations to appear on the same line as all job listings. This way, all possible matches were made, and the data was prepared for labeling.

www.review_and_job_matching .com



Data Labeling

In this project, we employed an automated labeling technique due to the large size of our dataset and its suitability for such methods. During the data labeling process, the following information was extracted and modeled:

- Education levels and their rankings (e.g., bachelor's, master's).
- Required experience ranges.
- Skill matching rates.
- Location compatibility.
- Work type and duration (e.g., remote work, full-time).

```python
# Eğitim seviyesi sıralaması
education_levels = ["ön lisans", "lisans", "yüksek lisans", "doktora"]

# Eğitim seviyesini sıralamada bulmak için yardımcı fonksiyon
def get_education_rank(education):
    if pd.isna(education):
        return -1  # Belirtilmemişse
    education_list = [ed.strip() for ed in str(education).split(",")]
    ranks = [education_levels.index(ed) for ed in education_list if ed in education_levels]
    return max(ranks, default=-1)  # En yüksek seviyesi

# Eğitim seviyesi kontrolü
def check_education(cv_education, job_education):
    cv_rank = get_education_rank(cv_education)
    job_rank = get_education_rank(job_education)

    if cv_rank == -1:
        return 0  # CV'de eğitim seviyesi belirtilmemişse
    if job_rank == -1 or cv_rank >= job_rank:
        return 1  # İş ilanında belirtilmemiş veya CV seviyesi yeterliyse
    return 0  # CV seviyesi iş ilanına uygun değilse
```

```python
# Çalışma yeri ve süresi eşleşmesi
def check_work_place_and_duration(job_row, cv_row):
    return job_row["Uzaktan/Normal"] == cv_row["Çalışma Şekli2"] and job_row["Çalışma Şekli"] == cv_row["Çalışma Türü"]

# Konum eşleşmesi
def check_location(job_row, cv_row):
    cv_location = cv_row["konum"]
    job_locations = str(job_row["Konum"]).split(",")  # İş ilanındaki iller virgülle ayrılmış olabilir
    return cv_location in job_locations
```

- **Full-time**
- **Part-time**
- **Contract**
- **Intern**

- **At Work**
- **Hybrid**
- **Remote**

```python
# Yetenekler eşleşme oranı
def check_skills(job_row, cv_row):
    job_skills = set(str(job_row["Yetenekler"]).split(","))
    cv_skills = set(str(cv_row["YETENEKLER"]).split(","))
    if not job_skills or not cv_skills:
        return 0  # Yeteneklerden biri belirtilmemişse eşleşme olmaz
    matching_skills = job_skills.intersection(cv_skills)
    return len(matching_skills) / len(job_skills)  # İş ilanındaki yeteneklerin ne kadarına uyuluyor
```

```python
# Tecrübe eşleşmesi
def check_experience(job_row, cv_row):
    job_experience = job_row["İstenen Tecrübe"]
    cv_experience = cv_row["ÇALIŞMA ZAMANI (YIL)"]

    if pd.isna(cv_experience):
        return False  # CV'de tecrübe belirtilmemişse uygun olmadığını varsayın

    if pd.isna(job_experience) or str(job_experience).strip() == "":
        return True  # İş ilanında tecrübe belirtilmemişse uygun olduğunu varsayın

    try:
        if " ile " in str(job_experience):  # Örnek: "5 ile 10"
            min_exp, max_exp = map(int, str(job_experience).split(" ile "))
            return min_exp <= cv_experience <= max_exp
        else:  # Tek bir sayı belirtilmişse
            min_exp = int(job_experience)
            return cv_experience >= min_exp
    except ValueError:
        return False  # Beklenmeyen bir format varsa uygun olmadığını varsayın
```

```python
# Ek özellik kontrolü (ÖNE ÇIKAN PROJE, SERTİFİKA ADI, GÖNÜLLÜLÜK YAPTIĞI ORGANİZASYON ADI)
def check_additional_features(cv_row):
    score = 0
    if not pd.isna(cv_row["ÖNE ÇIKAN PROJE"]) and str(cv_row["ÖNE ÇIKAN PROJE"]).strip():
        score += 1  # Proje varsa 1 puan
    if not pd.isna(cv_row["SERTİFİKA ADI"]) and str(cv_row["SERTİFİKA ADI"]).strip():
        score += 1  # Sertifika varsa 1 puan
    if not pd.isna(cv_row["GÖNÜLLÜLÜK YAPTIĞI ORGANİZASYON ADI"]) and str(cv_row["GÖNÜLLÜLÜK YAPTIĞI ORGANİZASYON ADI"]).strip():
        score += 1  # Gönüllülük varsa 1 puan
    return score / 3  # Bu özelliklerin toplamı, eşleşme oranına katkıda bulunur (0 ile 1 arasında)

# Eşleşmeyi kontrol et
def calculate_match_percentage(job_row, cv_row):
    total_checks = 6
    matches = 0

    if check_work_place_and_duration(job_row, cv_row):
        matches += 1
    if check_location(job_row, cv_row):
        matches += 1
    if check_experience(job_row, cv_row):
        matches += 1
    if check_education(cv_row["DERECE"], job_row["Eğitim Seviyesi"]):
        matches += 1
    matches += check_skills(job_row, cv_row)  # Yetenekler eşleşme oranı (0 ile 1 arasında)
    matches += check_additional_features(cv_row)  # Ek özellikler eşleşme oranı (0 ile 1 arasında)

    return (matches / total_checks) * 100 if total_checks > 0 else 0
```

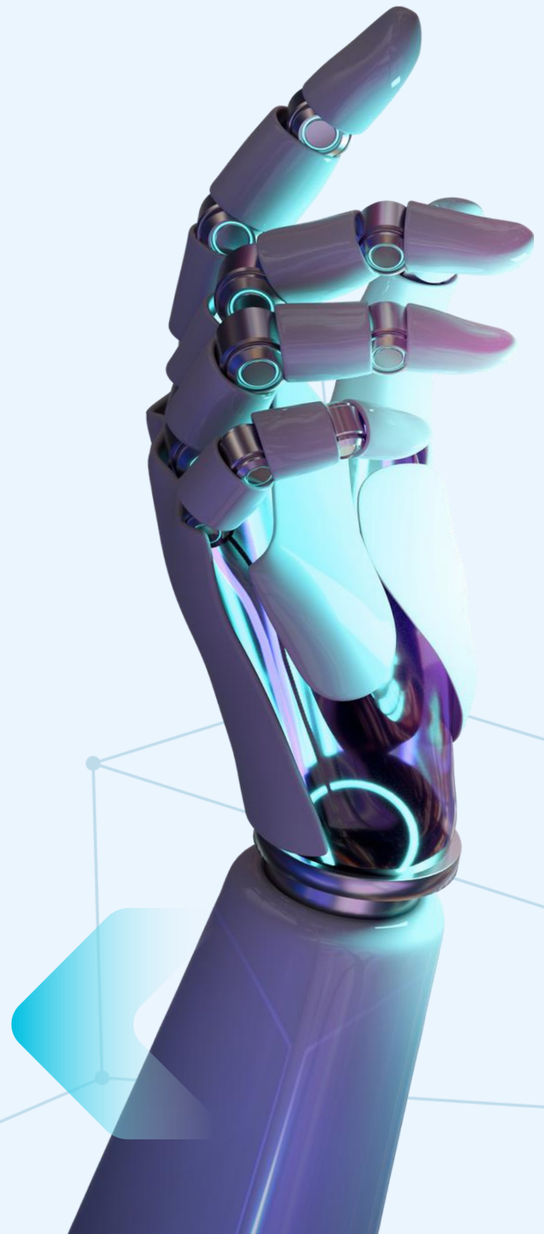
Model Training

Used Models: Logistic Regression, LightGBM, Random Forest

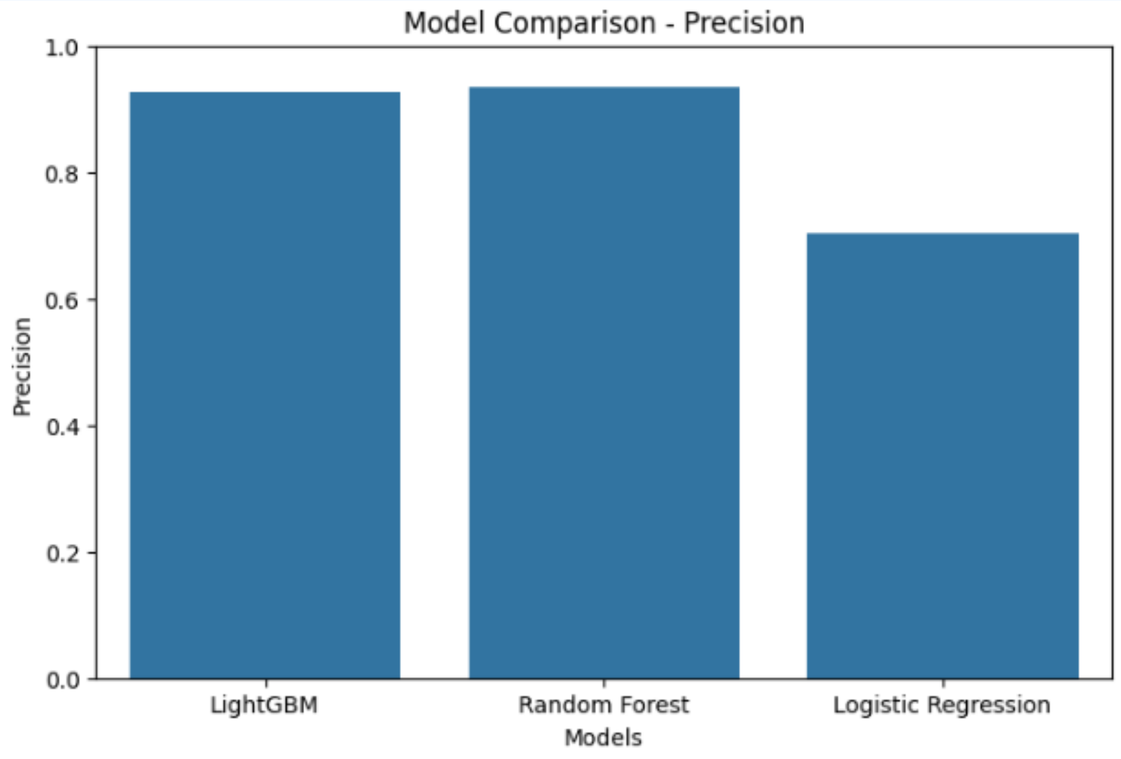
- **Logistic Regression:** Simple, interpretable; suitable for linear relationships
- **LightGBM:** Efficient with large datasets, strong performance for classification
- **Random Forest:** Improves prediction accuracy through ensemble learning

Performance Metrics: Accuracy, precision, recall, F1 score

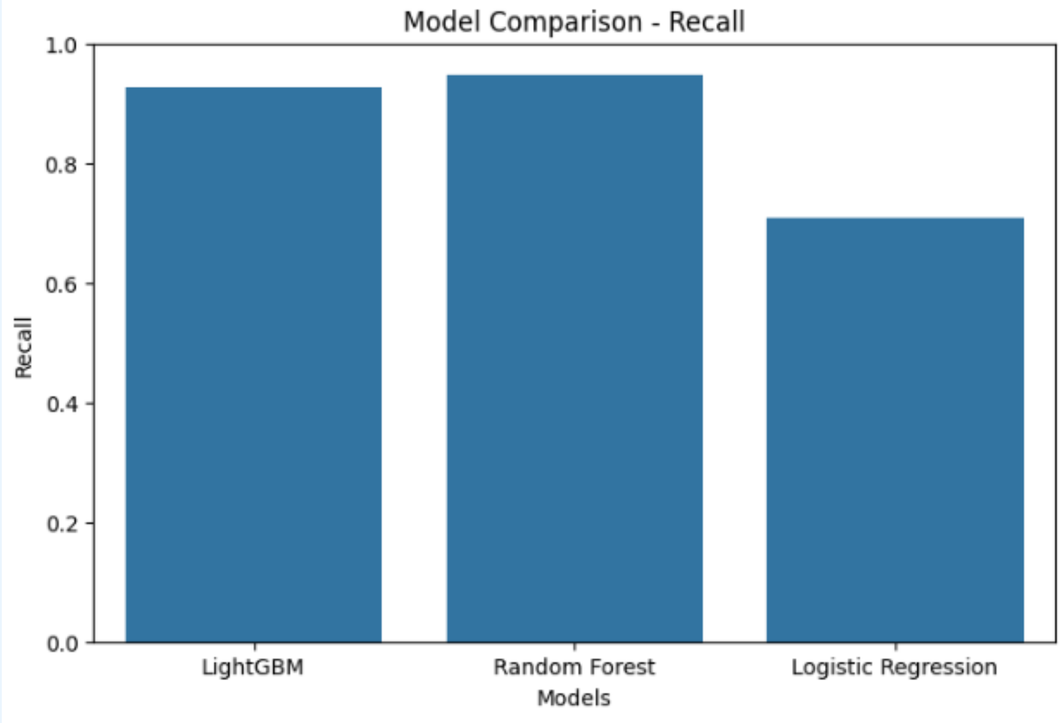
The best model was selected and deployed based on **cross-validation** results



Results

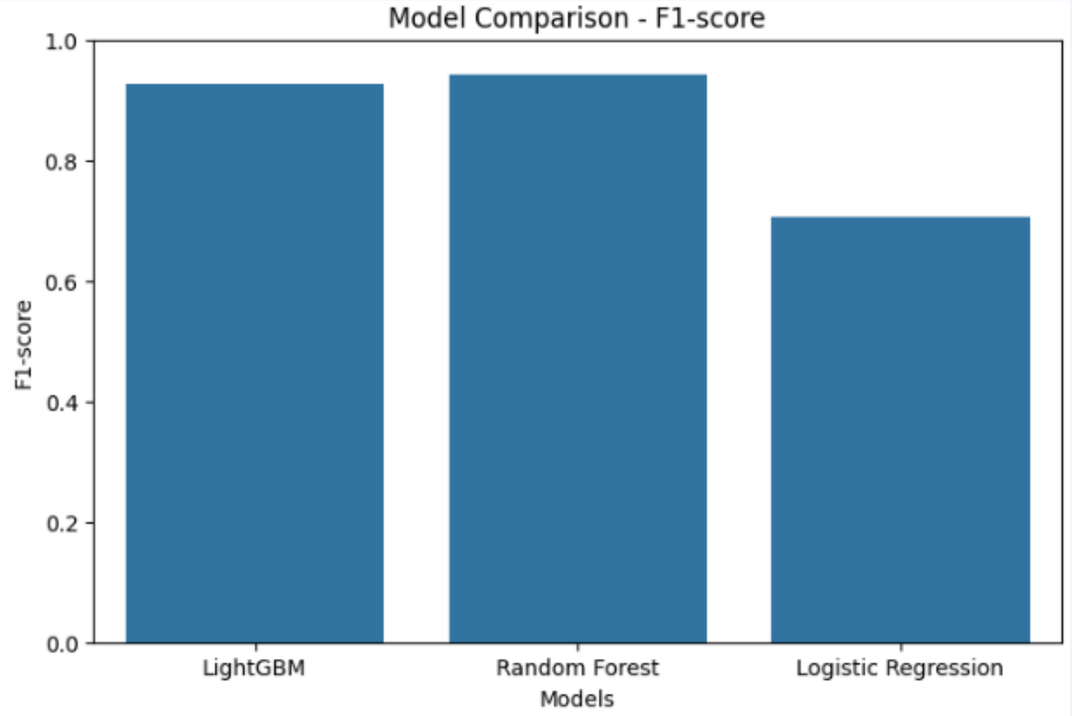


Precision Scores:
LightGBM: 0.9267
Random Forest: 0.9354
Logistic Regression: 0.7025



Recall Scores:
LightGBM: 0.9275
Random Forest: 0.9466
Logistic Regression: 0.7087

Results



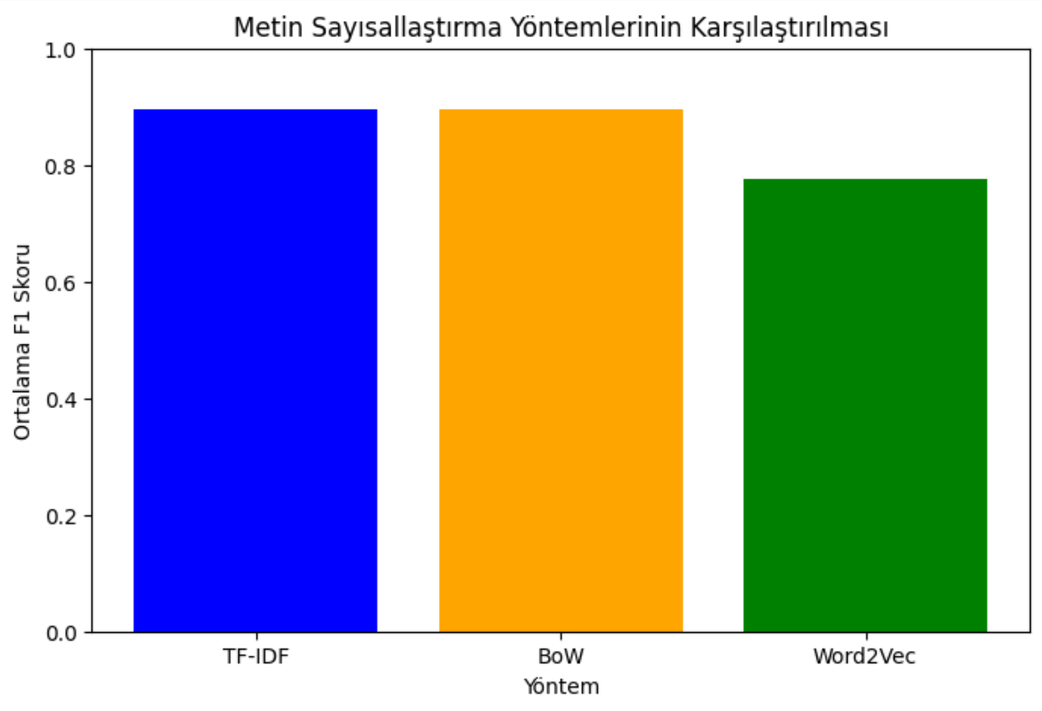
F1-score Scores:
LightGBM: 0.9271
Random Forest: 0.9410
Logistic Regression: 0.7056

MODEL	FEATURE EXTRACTION	EVALUATION METHOD	ACCURACY	PRECISION	RECALL	F1-SCORE
LightGBM	TF-IDF	Cross Validation	-	-	-	0.8947
LightGBM	BoW	Cross Validation	-	-	-	0.8966
LightGBM	Word2Vec	Cross Validation	-	-	-	0.7770
LightGBM	TF-IDF	Train-Test Split	0.8977	0.9209	0.8677	0.8935
LightGBM	BoW	Train-Test Split	0.9024	0.9278	0.8704	0.8982
LightGBM	Word2Vec	Train-Test Split	0.7718	0.7585	0.7903	0.7741
Logistic Regression	TF-IDF	Cross Validation	-	-	-	0.6839
Logistic Regression	BoW	Cross Validation	-	-	-	0.6855
Logistic Regression	Word2Vec	Cross Validation	-	-	-	0.6637
Logistic Regression	TF-IDF	Train-Test Split	0.71395	0.708767	0.715860	0.712296
Logistic Regression	BoW	Train-Test Split	0.71560	0.707983	0.723441	0.715628
Logistic Regression	Word2Vec	Train-Test Split	0.67630	0.668308	0.686142	0.677107
Random Forest	TF-IDF	Cross Validation	-	-	-	0.8435
Random Forest	BoW	Cross Validation	-	-	-	0.8424
Random Forest	Word2Vec	Cross Validation	-	-	-	0.7442
Random Forest	TF-IDF	Train-Test Split	0.94300	0.965437	0.917619	0.940920
Random Forest	BoW	Train-Test Split	0.94955	0.970850	0.925806	0.947793
Random Forest	Word2Vec	Train-Test Split	0.82060	0.812407	0.828667	0.820456
Naive Bayes	TF-IDF	Cross Validation	-	-	-	0.6120
Naive Bayes	BoW	Cross Validation	-	-	-	0.6509
Naive Bayes	Word2Vec	Cross Validation	-	-	-	0.6049
Naive Bayes	TF-IDF	Train-Test Split	0.6299	0.6357	0.5897	0.6118
Naive Bayes	BoW	Train-Test Split	0.6490	0.6390	0.6675	0.6530
Naive Bayes	Word2Vec	Train-Test Split	0.6126	0.6145	0.5816	0.5976
LSTM	TF-IDF	Train-Test Split	0.9389	0.9537	0.9214	0.9372
LSTM	BoW	Train-Test Split	0.9526	0.9753	0.9275	0.9508
LSTM	Word2Vec	Train-Test Split	0.9397	0.9457	0.9316	0.9380

LightGBM

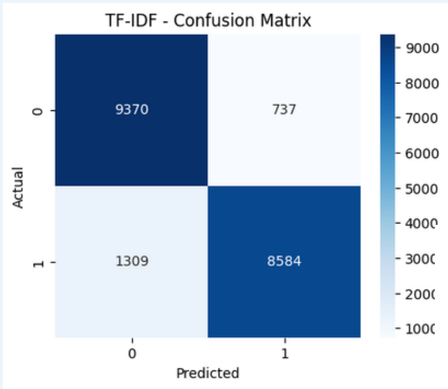
Using lightGBM as an algorithm, TF-IDF, Bag of Words and Word2Vec were compared for Feature Extraction.

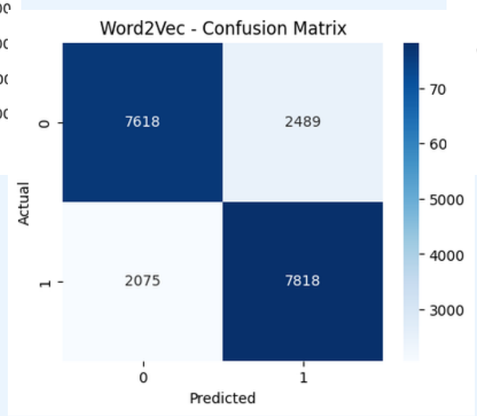
Cross validation

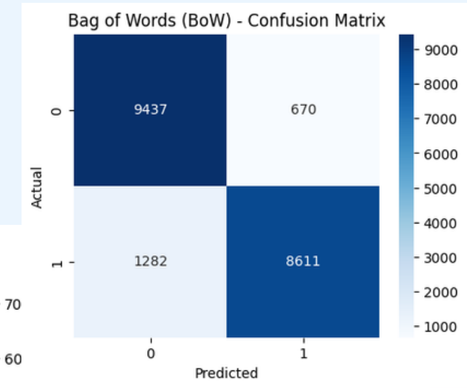


TF-IDF - Cross Validation Mean F1 Score: 0.8947
BoW - Cross Validation Mean F1 Score: 0.8966
Word2Vec - Cross Validation Mean F1 Score: 0.7770

Train test split





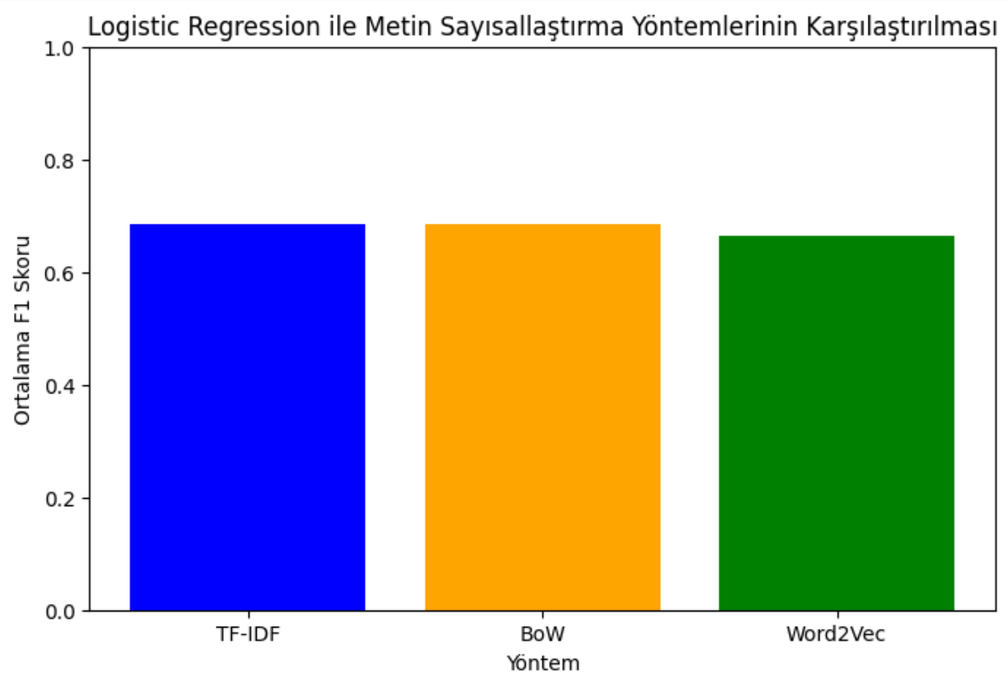


TF-IDF Performance:
Accuracy: 0.8977, Precision: 0.9209, Recall: 0.8677, F1-Score: 0.8935
Bag of Words (BoW) Performance:
Accuracy: 0.9024, Precision: 0.9278, Recall: 0.8704, F1-Score: 0.8982
Word2Vec Performance:
Accuracy: 0.7718, Precision: 0.7585, Recall: 0.7903, F1-Score: 0.7741

Logistic Regression

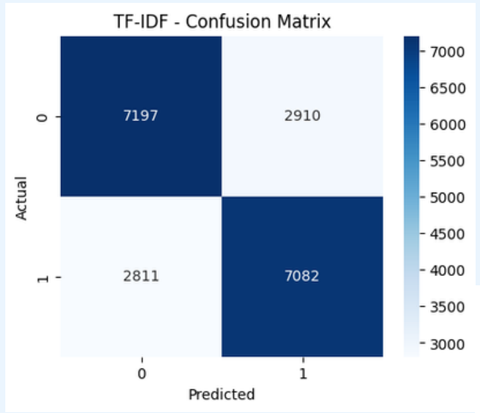
Using Logistic Regression as an algorithm, TF-IDF, Bag of Words and Word2Vec were compared for Feature Extraction.

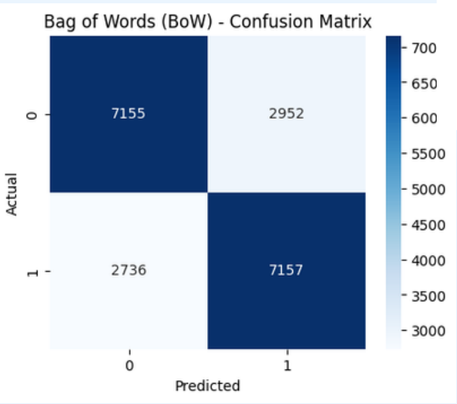
Cross validation

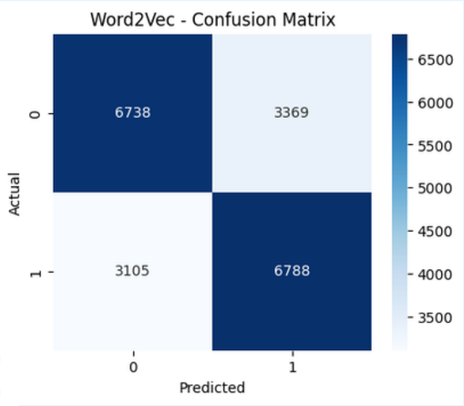


TF-IDF - Cross Validation Mean F1 Score: 0.6839
BoW - Cross Validation Mean F1 Score: 0.6855
Word2Vec - Cross Validation Mean F1 Score: 0.6637

Train test split





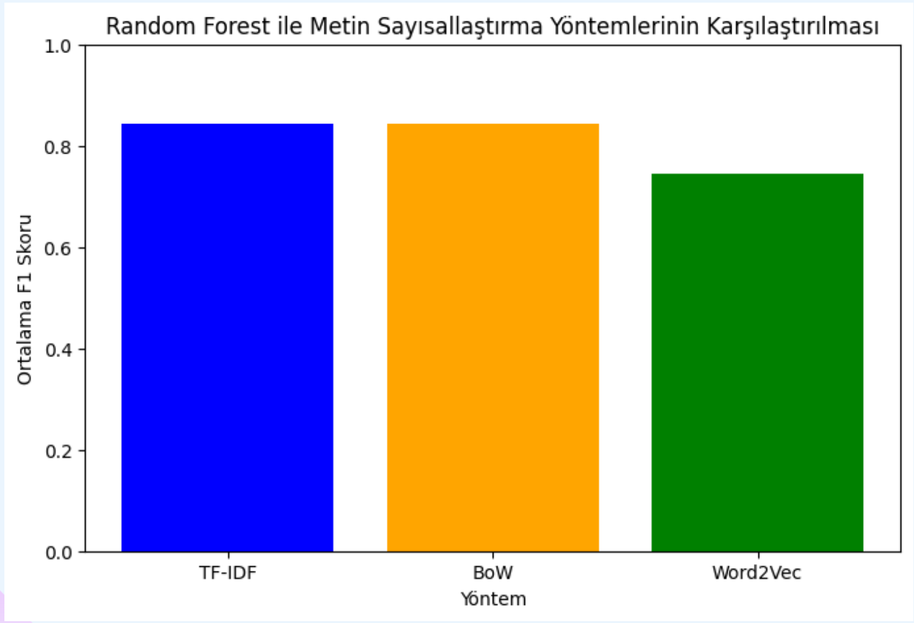


	Method	Accuracy	Precision	Recall	F1-Score
0	TF-IDF	0.71395	0.708767	0.715860	0.712296
1	Bag of Words (BoW)	0.71560	0.707983	0.723441	0.715628
2	Word2Vec	0.67630	0.668308	0.686142	0.677107

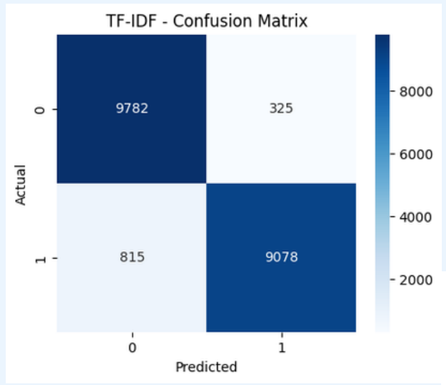
Random Forest

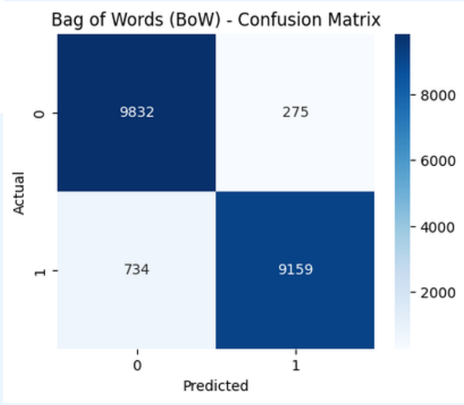
Using Random Forest as an algorithm, TF-IDF, Bag of Words and Word2Vec were compared for Feature Extraction.

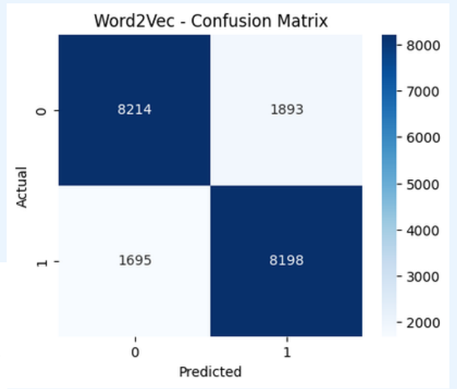
Cross validation



Train test split







TF-IDF - Cross Validation Mean F1 Score: 0.8435
BoW - Cross Validation Mean F1 Score: 0.8424
Word2Vec - Cross Validation Mean F1 Score: 0.7442

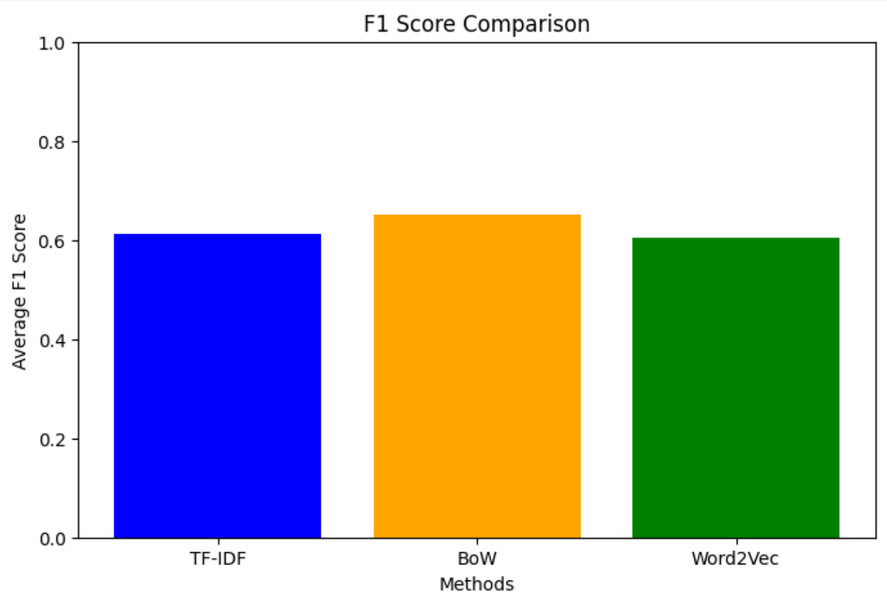
Değerlendirme Sonuçları:

	Method	Accuracy	Precision	Recall	F1-Score
0	TF-IDF	0.94300	0.965437	0.917619	0.940920
1	Bag of Words (BoW)	0.94955	0.970850	0.925806	0.947793
2	Word2Vec	0.82060	0.812407	0.828667	0.820456

Naive Bayes

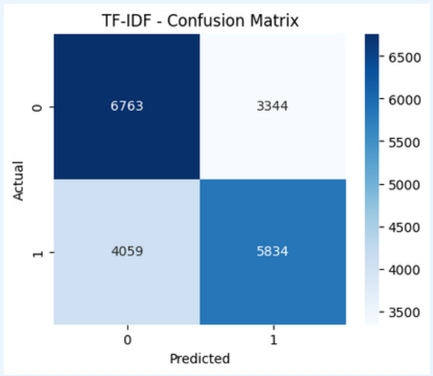
Using Naive Bayes as an algorithm, TF-IDF, Bag of Words and Word2Vec were compared for Feature Extraction.

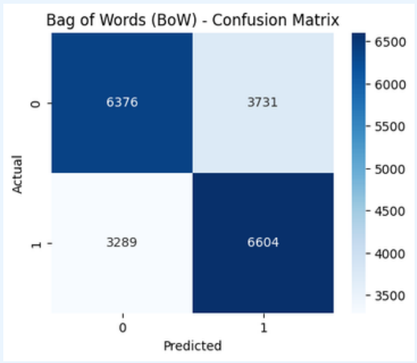
Cross validation

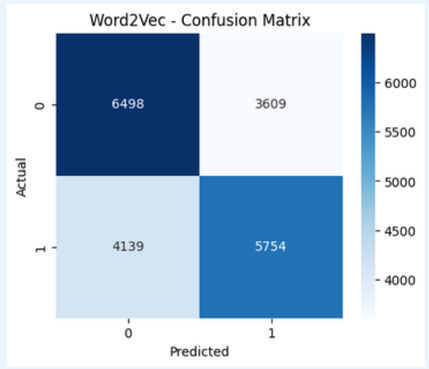


TF-IDF - Average F1 Score: 0.6120
Bag of Words (BoW) - Average F1 Score: 0.6509
Word2Vec - Average F1 Score: 0.6049

Train test split



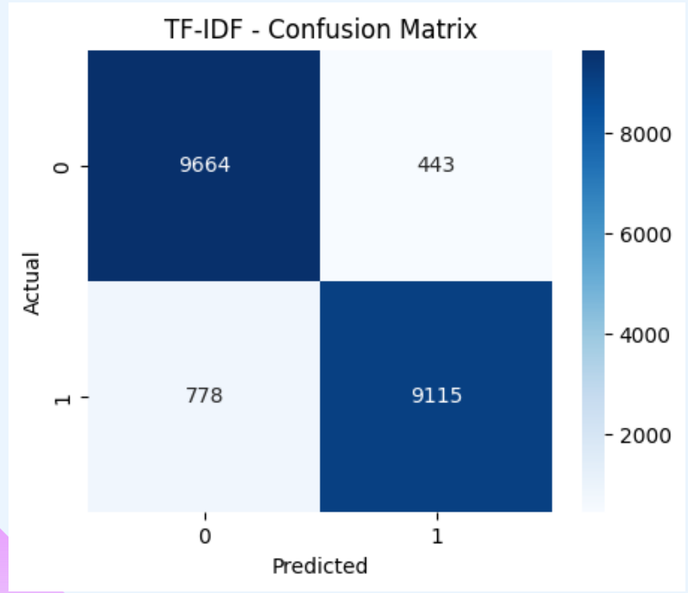


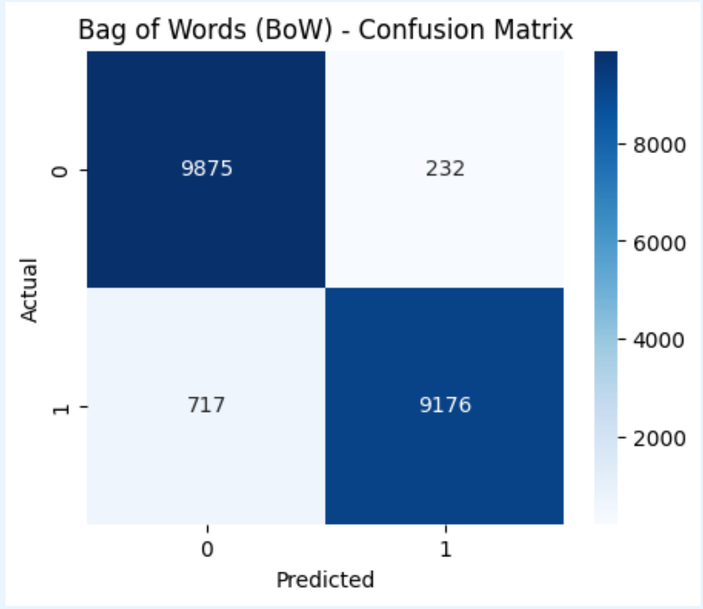


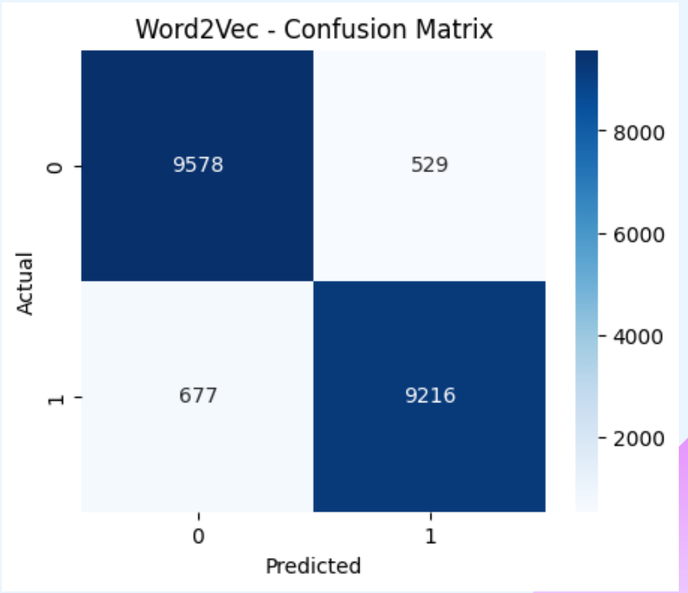
TF-IDF Performance:
Accuracy: 0.6299, Precision: 0.6357, Recall: 0.5897, F1-Score: 0.6118
Bag of Words (BoW) Performance:
Accuracy: 0.6490, Precision: 0.6390, Recall: 0.6675, F1-Score: 0.6530
Word2Vec Performance:
Accuracy: 0.6126, Precision: 0.6145, Recall: 0.5816, F1-Score: 0.5976

LSTM

Using Naive Bayes as an algorithm, TF-IDF, Bag of Words and Word2Vec were compared for Feature Extraction.







TF-IDF Performance:
Accuracy: 0.9389, Precision: 0.9537, Recall: 0.9214, F1-Score: 0.9372

Bag of Words (BoW) Performance:
Accuracy: 0.9526, Precision: 0.9753, Recall: 0.9275, F1-Score: 0.9508

Word2Vec Performance:
Accuracy: 0.9397, Precision: 0.9457, Recall: 0.9316, F1-Score: 0.938

THANK YOU!