# بسم الله الرحمن الرحيم


# گزارش کار پروژه‌ی داده‌کاوی

**دانشجو:**

**علی بدیعی گورتی**

**بهمن ماه ۱۴۰۲**

## فهرست

## چکیده

پروژه داده کاوی: ارزیابی سختی سوالات با استفاده از مدل‌های متنوع

در این پروژه، از تنوع تکنیک‌های داده کاوی و یادگیری ماشین بهره گرفته می‌شود تا اطلاعات از یک سایت جمع‌آوری شده و ویژگی‌های متنوع و حیاتی از سوالات استخراج گردد. از الگوریتم‌های معتبر و متنوعی از جمله SVM، XGBoost، Naive Bayes، SVC، CART، Logistic Regression و MLP برای آموزش مدل‌های داده‌کاوی بهره می‌گیریم. به عنوان یک الگوریتم پیشرفته بردار پشتیبانی، توانمندی بسیاری در دسته‌بندی دارد و از آن برای تجزیه و تحلیل ویژگی‌ها بهره می‌بریم. XGBoost با قابلیت گرادیان بوستینگ و درخت تصمیم، در افزایش دقت و کارایی مدل‌ها به‌کار گرفته می‌شود. با ترکیب این الگوریتم‌ها و اجرای آموزش بر روی داده‌های آموزشی، دانشجوان می‌توانند به تحلیل گسترده‌تری از عوامل موثر بر سختی سوالات دست پیدا کنند. این پروژه فراهم‌کننده فرصتی مناسب برای توسعه مهارت‌های تحلیل داده، استفاده از تکنیک‌های پیشرفته داده‌کاوی، و ایجاد مدل‌های پیش‌بینی بر اساس داده‌های واقعی است. این تجربه به دانشجوان این امکان را می‌دهد که به عنوان تحلیلگران داده، در حوزه تحلیل داده‌های پیچیده و متنوع تخصص کسب کنند.

**مقدمه**

در فصل اول، به بررسی پیش‌نیازهای برنامه‌نویسی و علمی می‌پردازیم.

در فصل دوم، به اعمال انجام شده اشاره شده است. تمامی مراحل تبدیل فایل های train.csv و valid.csv به موارد خواسته شده در متن توضیح داده شده و نحوه پردازش روی متن نیز توضیح داده شده است. سپس نحوه‌ی ساخت داده‌های Train و Test را تشریح کرده و در نهایت، پیاده‌سازی و آموزش مدل‌های Naïve Bayes، SVC، CART، Logistic Regression، MLP و XGBoost را بیان کرده‌ایم. خروجی‌های این فاز برای تحلیل در فصل سوم مورد استفاده قرار گرفته‌اند.

در فصل سوم، نتایج مورد بررسی قرار گرفته و مدل‌ها با یکدیگر مقایسه می‌شوند.

## فصل اول: مقدمه به پیش‌نیازهای برنامه‌نویسی و علمی

در این فصل، به بررسی و توضیح پیش‌نیازهایی که برنامه‌نویسان و علمای داده باید درک کنند، می‌پردازیم. این پیش‌نیازها از جمله مفاهیم اساسی برنامه‌نویسی، الگوریتم‌ها، و مهارت‌های علمی شامل تجزیه و تحلیل داده، استخراج اطلاعات مفید، و تفسیر نتایج آماری می‌شوند. هدف این فصل، فراهم کردن زمینه‌ای مناسب برای درک بهتر مفاهیم و ابزارهایی است که در فازهای بعدی پروژه به کار گرفته می‌شوند. به علاوه، توضیحاتی در مورد استانداردها و روش‌های مرتبط با برنامه‌نویسی و علم داده نیز ارائه خواهد شد.

### پیش‌نیازهای برنامه‌نویسی برای انجام این پروژه:

۱. مهارت در زبان‌های برنامه‌نویسی:

برنامه‌نویسی یکی از مهارت‌های اساسی است که در این پروژه به طور فعال به کار خواهد رفت. مهارت در زبان‌هایی مانند Python یا R امکان اجرای کدها و پردازش داده‌ها را بهبود می‌بخشد. در این پروژه از زبان پایتون استفاده شده است.

۲. آشنایی با مفاهیم پایگاه داده:

اطلاعات مورد نیاز برای پروژه از پایگاه داده‌های Stack Exchange به دست می‌آید. بنابراین، آشنایی با مفاهیم مانند SQL و نحوه‌ی استخراج داده از پایگاه داده‌ها از اهمیت بالایی برخوردار است.

۳. توانایی در تحلیل داده:

تحلیل داده‌ها و استفاده از روش‌های مختلف برای استخراج الگوها و اطلاعات از داده‌های حاصل از پایگاه داده Stack Exchange ضروری است. مفاهیم مانند خوشه‌بندی، تجزیه و تحلیل آماری، و تصویرسازی داده می‌توانند مفید باشند.

۴. آشنایی با مفاهیم داده‌کاوی:

در فازهای مختلف پروژه، داده‌ها نیاز به برچسب‌زنی و دسته‌بندی دارند. آشنایی با مفاهیم داده‌کاوی و تکنیک‌های مختلف برچسب‌زنی و دسته‌بندی می‌تواند در اینجا مفید باشد.

۵. مهارت در استفاده از ابزارهای مرتبط:

استفاده از ابزارهایی مانند Jupyter Notebook و Pycharm برای اجرای کد، ایجاد گزارش‌های تحلیلی، و ایجاد نمودارها به اهمیت زیادی دارد. مهارت در استفاده از این ابزارها به بهبود فرآیند تحلیل و گزارش‌دهی کمک خواهد کرد.

با توجه به این پیش‌نیازها، برنامه‌نویسان می‌بایست با دقت و توجه به جزئیات این مراحل را پیش ببرند تا به بهترین نتایج در انجام پروژه دست یابند.

## پیش‌نیازهای علمی برای انجام این پروژه:

۱. دانش در زمینه‌ی معیارهای ارزیابی:

در تحلیل داده‌ها و پیاده‌سازی مدل‌های یادگیری ماشین، شناخت دقیق از معیارهای ارزیابی از جمله F1 Score و Accuracy اساسی است. فهم درست از این معیارها در ارزیابی عملکرد مدل‌ها حائز اهمیت است.

۲. آشنایی با تکنیک‌های برچسب‌زنی و دسته‌بندی:

برچسب‌زنی داده‌ها به کمک مدل‌های یادگیری ماشین نیاز به فهم عمیق از تکنیک‌های برچسب‌زنی دارد. در اینجا، مفاهیمی مانند ماتریس درهم‌ریختگی (Confusion Matrix) می‌توانند به درک بهتری از عملکرد مدل‌ها کمک کند.

۳. آشنایی با تحلیل دقیق نتایج آماری:

در فازهای مختلف پروژه، نیاز به تحلیل دقیق نتایج آماری و اعتبارسنجی مدل‌ها وجود دارد. آشنایی با مفاهیم مانند انحراف معیار، p-value، و تفسیر نتایج آماری می‌تواند در اینجا مفید باشد.

۴. مهارت در ایجاد و بهینه‌سازی مدل‌های یادگیری ماشین:

توانایی در پیاده‌سازی و بهینه‌سازی مدل‌های یادگیری ماشین مهارتی حیاتی است. این شامل انتخاب و تنظیم پارامترها، تجزیه و تحلیل خطاها، و بهبود عملکرد مدل‌ها می‌شود.

۵. آشنایی با اصول علم داده:

اصول علم داده از قبیل تجزیه و تحلیل داده‌ها، استخراج ویژگی‌ها، و پیش‌پردازش داده‌ها در فازهای مختلف این پروژه اهمیت زیادی دارد. نیاز به درک اصول علم داده برای بهترین استفاده از داده‌ها و حصول اطلاعات مفید وجود دارد.

این پیش‌نیازها با همکاری موازی از برنامه‌نویسی و مفاهیم علم داده به طور کامل به انجام موثر پروژه کمک می‌کنند.

## فصل دوم: تحلیل، طراحی و اجری فاز

در فصل‌های گذشته، ما به مرحله‌های جمع‌آوری داده‌ها، پیش‌پردازش آن‌ها و برچسب‌زنی اطلاعات پرداختیم. حالا، با دستیابی به یک دیتاست کامل و آماده، در فصل چهارم به تحلیل و پردازش داده‌ها به صورت جزئیات می‌پردازیم. در این فصل، ابتدا به تبدیل خروجی فاز دو به موارد خواسته شده در متن پروژه می‌پردازیم. این مرحله شامل تحلیل و استخراج ویژگی‌های مهم از داده‌ها، اطلاعات توافق و عدم توافق در ارزیابی‌ها، و سایر موارد تحلیلی است.

پس از تبدیل خروجی فاز دو، به پردازش متن سوالات می‌پردازیم و تحلیل روی آن‌ها انجام می‌دهیم. از تکنیک‌های پردازش متن استفاده می‌کنیم تا الگوها و اطلاعات ارزشمندی را از متون سوالات استخراج کنیم.

یکی از بخش‌های حیاتی این فصل، تبدیل متون توضیحات به نمودارهای عددی و آماده‌سازی داده‌ها برای آموزش مدل‌های یادگیری ماشین است. با استفاده از توابعی چون Naïve Bayes، SVC، CART، Logistic Regression، MLP و XGBoost، به آموزش مدل‌ها می‌پردازیم.

نهایتاً، خروجی‌های این فصل را برای مقایسه مدل‌ها و تفسیر نتایج به فصل بعد منتقل می‌کنیم.

در این مرحله از پروژه، نیاز داریم یک متن واحد از محتوای موجود در ستون‌های Body، Title و Tags به دست آورده و در یک ستون جدید با نام MergedText ذخیره کنیم:

```
# Merge QuestionTags, QuestionTitle and QuestionBody fields into a new feature 'MergedText'
df['MergedText'] = (df['Tags'].astype(str) + ', ' + df['Title'].astype(str) + ', ' +
                    df['Body'].astype(str))
```

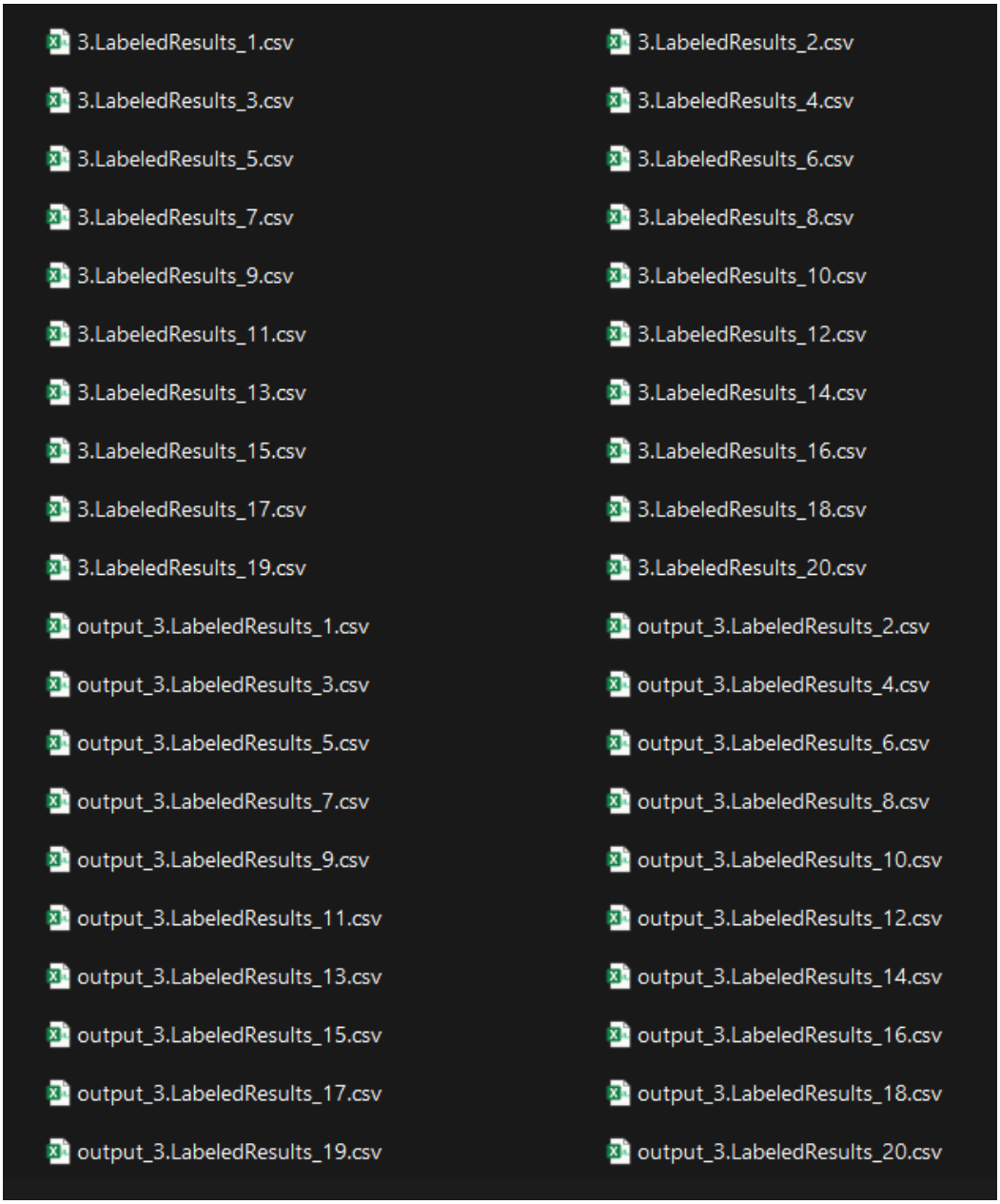
سپس، برای هر داده در ستون MergedText، عملیات preprocess را روی آن اعمال کرده و نتیجه را در یک ستون جدید به نام ProcessedText ذخیره می کنیم:

```
# Apply text preprocessing to the 'MergedText' column
df['ProcessedText'] = tqdm(
    df['MergedText'].progress_apply(self.preprocess_given_text, stop_words=stop_words, ps=ps),
    total=len(df))
```

**نکته:**

در ابتدای پروژه، از دیتاست تستی استفاده میشد. در انتها امکان عوض کردن دیتاست اصلی با دیتاست تستی ممکن نبود. این به دلیل این بود که دیتاست تستی تنها دارای ۱۰ داده بود ولی دیتاست اصلی شامل ۱۰۰۰ داده بود. انجام عملیات preprocessing روی تعداد زیادی داده به طور مستقیم زمان بسیار زیادی را می‌طلبید و همچنین حجم قابل توجهی از حافظه RAM را اشغال می کرد. برای حل این مشکلات، توابعی به برنامه افزوده شدند که با استفاده از تکنیک‌های برنامه‌نویسی، این مشکلات را حل کنند. یکی از این تکنیک‌ها، استفاده از روش تقسیم و غلبه بود که فایل CSV اصلی را به چندین بخش تقسیم کرد. هر بخش تعداد داده یکسانی را شامل می‌شد. سپس عملیات preprocessing بر روی هر بخش اعمال شد و در نهایت، تمام بخش‌ها با یکدیگر ادغام شدند. این کار با استفاده از توابع split_csv و merge_csv انجام شد. همچنین برای بهبود قابلیت‌ها و زیبایی توابع، شمارنده و progressbar به آن‌ها افزوده شد. در این روند، فایل اصلی ابتدا به ۲۰ فایل فرعی (در دایرکتوری TextProcessedFiles) تقسیم شد. سپس، هر یک از این فایل‌ها به ترتیب بررسی شد و خروجی‌ها در فایل‌های جدیدی ذخیره شدند.

استفاده از فایل‌های فرعی در اینجا به این معناست که هر فایل به طور مستقل بررسی و پردازش می‌شود. این رویکرد این امکان را فراهم می‌کند که در صورت بروز هر گونه خطا یا بسته شدن ناگهانی برنامه، تنها فایل در حال پردازش از دست برود و داده‌های دیگر از بین نرود. این مزیت به اجرای مطمئن‌تر و ادامه‌پذیرتر عملیات preprocessing کمک می‌کند.



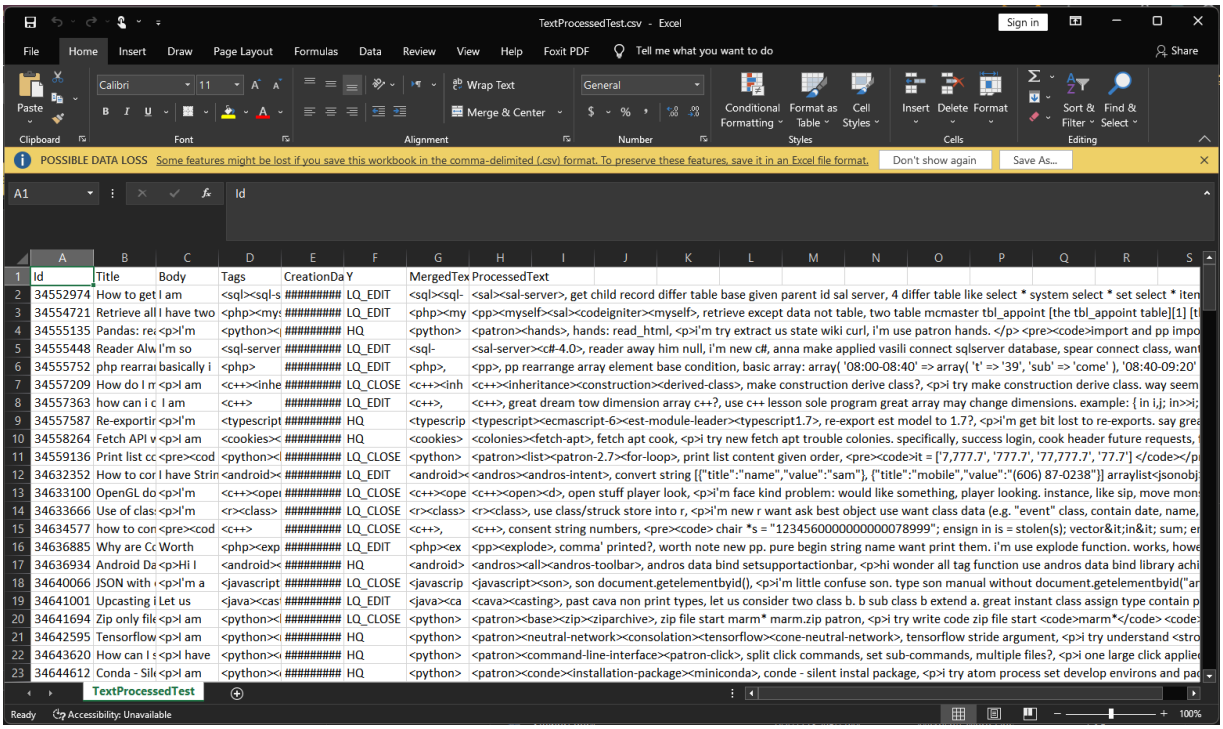

فایل های ساخته شده در این مرحله (خروجی ها با output شروع میشوند)

```
# Remove stop words
words = [word for word in text.lower().split() if word.lower() not in stop_words]

# Perform stemming
stemmed_words = [ps.stem(word) for word in words]

# Correct spellings using TextBlob
corrected_text = ' '.join([str(TextBlob(word).correct()) for word in stemmed_words])
```

پیش پردازش متن (شامل حذف stop words، تصحیح غلط املایی و ...)

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Master\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

Start splitting the SCV !
Finish splitting the SCV !
Start preprocessing on each file >>
(1/40) 3.LabeledResults_1.csv:
  6%|▌                                                  | 3/49 [00:24<06:12,  8.09s/it]
```

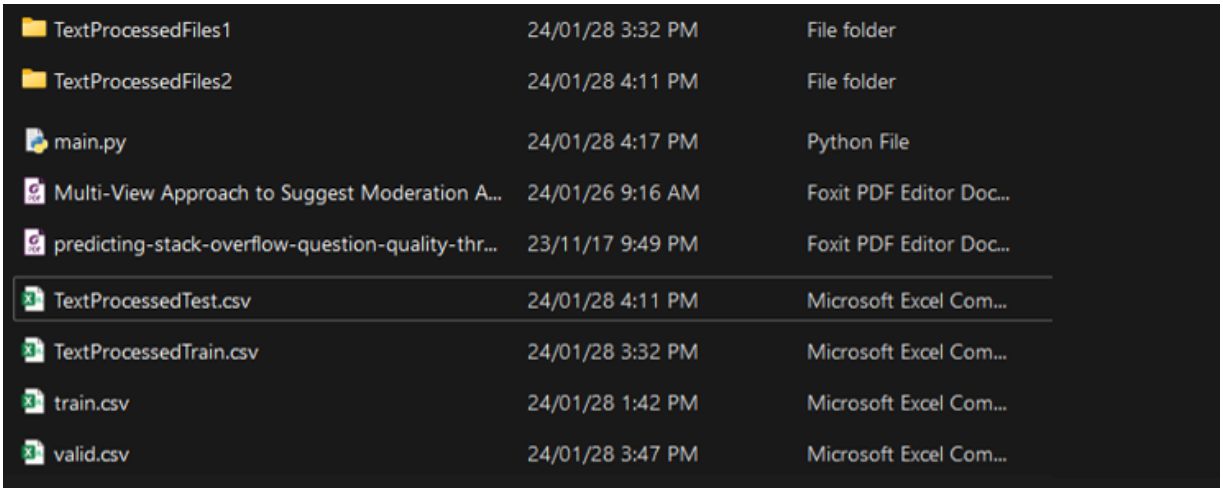
نمونه اجرای برنامه با اضافه شدن progressbar




داده های جدید (شامل متن پردازش شده و متن سرهم شده)

نکته:

فایل حاصل در این مرحله دارای حجم بسیار بزرگ است، به بیش از ۶۰۰ مگابایت می‌رسد. به عبارت دیگر، داده‌ها تا این نقطه به صورت زیر فراهم شده‌اند:




فایل های موجود تا این مرحله

---

برای بخش بعدی از پروژه، این داده‌ها قابل استفاده نیستند، زیرا حاوی حجم بسیار بالایی هستند. به منظور مدیریت بهتر حافظه RAM و کنترل مقدار ورودی، از تکنیک‌های برنامه‌نویسی استفاده می‌شود. به عنوان مثال، برای خواندن فایل CSV از ورودی، پارامتر "low_memory" به آن اضافه می‌شود:

```
# Load the CSV file into a DataFrame
df = pd.read_csv(input_file, low_memory=False)
```

---

در بخش سوم این فاز، فایل‌های آموزشی و آزمون از ورودی خوانده می‌شوند. سپس داده‌های تکراری، خالی و غیرقابل استفاده از آن حذف می‌شود. سپس "x" و "y" طبق ستون‌های "ProcessedText" و "QuestionLabel" جدا شده و از آن‌ها داده‌های آزمون و تست خوانده می‌شوند:

```
# Separate features and labels
x_train, y_train = train_df['ProcessedText'], train_df['QuestionLabel']
x_test, y_test = test_df['ProcessedText'], test_df['QuestionLabel']
```

برای بعضی از مدل ها، نیاز هست که متن توسط TF-IDF وکتورایز شود:

```
# Vectorize the text data using TF-IDF
vectorizer = TfidfVectorizer()
x_train_tfidf = vectorizer.fit_transform(x_train)
x_test_tfidf = vectorizer.transform(x_test)
```

سپس در این مرحله، مدل‌ها را تعریف می‌کنیم و برای هرکدام، داده‌ها را روی آن‌ها اعمال کرده و آموزش می‌دهیم. سپس مجموعه آزمون را روی آن تست می‌کنیم. پس از این مرحله، با استفاده از توابع آماده کتابخانه‌های مربوط، موارد خواسته‌شده از جمله دقت، صحت، F1 Score و Recall را محاسبه می‌کنیم:

```
# Initialize the classifiers
models = {
    'Naive Bayes': MultinomialNB(),
    'SVC': SVC(),
    'CART': DecisionTreeClassifier(),
    'Logistic Regression': LogisticRegression(),
    'MLP': MLPClassifier(max_iter=500),
    'XGBoost': xgb.XGBClassifier()
}
```

سپس نتایج را در یک دیکشنری ذخیره می‌کنیم. از دیکشنری ساخته‌شده، یک دیتافریم جدید ایجاد می‌کنیم. سپس با استفاده از کد مربوطه، ماتریس ابهام (confusion matrix) را برای هرکدام از مدل‌ها می‌سازیم و آن‌ها را کنار یکدیگر در یک فایل ذخیره می‌کنیم:

```python
# Train and evaluate each model
results = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1 Score': []}

for model_name, model in models.items():...

# Display the results
results_df = pd.DataFrame(results)
```

اعمال مدل ها

```python
# Plotting confusion matrix
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 15))

for ax, (model_name, model) in zip(axes.flatten(), models.items()):...

plt.tight_layout()
plt.savefig(confusion_matrix_name)
plt.show()
```

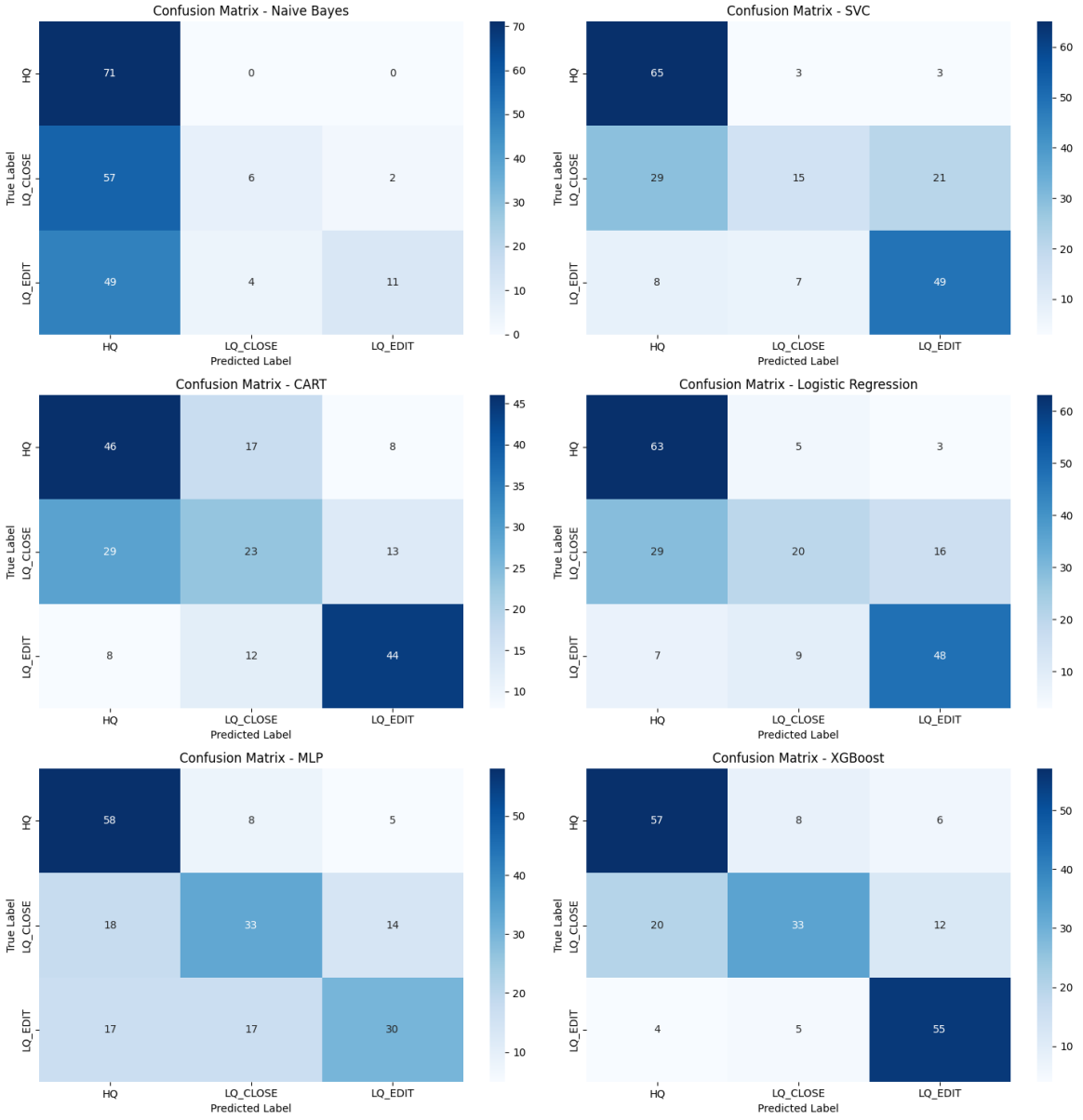
ماتریس confusion

سپس نمودار های مقایسه برای هر کدام از ویژگی های گفته شده در بالا، ساخته و آن ها را ذخیره میکنیم.

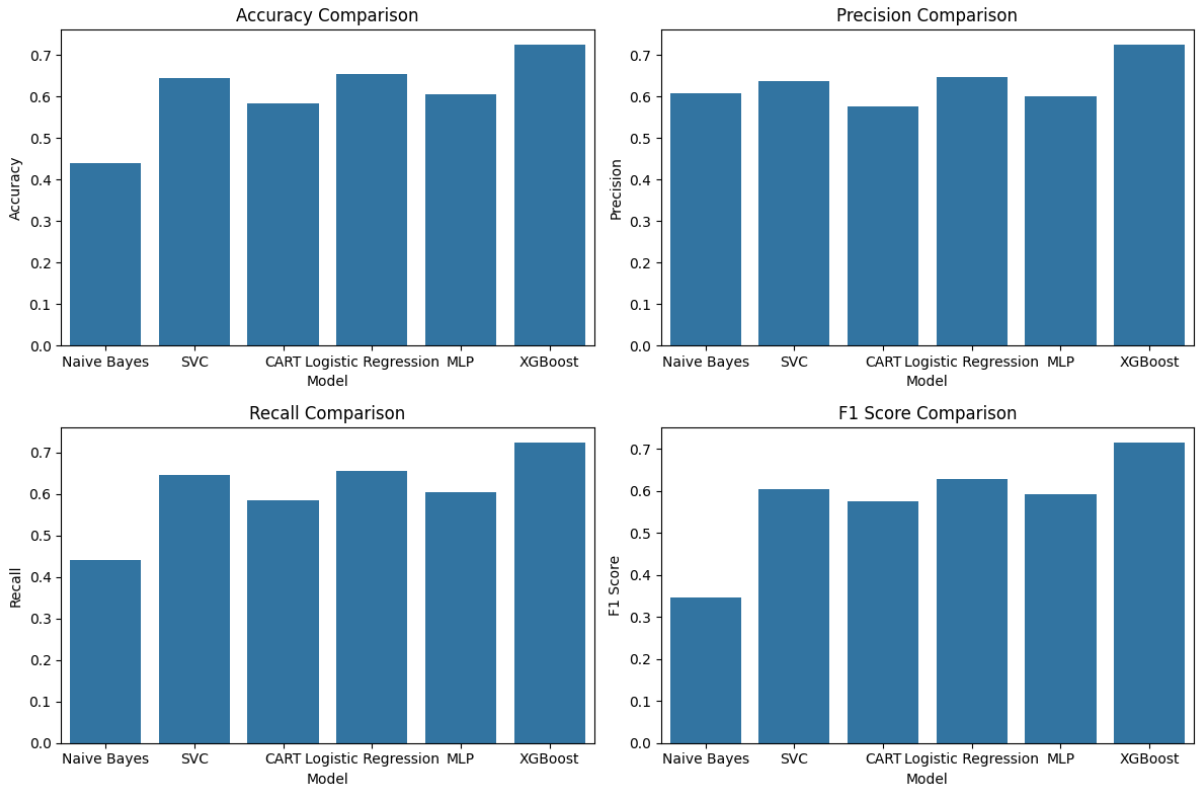
## فصل سوم: ارزیابی و بهبود نتایج

در فصل پنجم این پروژه، به مرحله‌ی ارزیابی و بهینه‌سازی مدل‌های آموزش دیده بر پایه دیتاهای آزمون می‌پردازیم. در این فاز، ابتدا مدل‌های پیش‌بینی طراحی شده را به عنوان یک مجموعه از دیتاهای آزمون مورد ارزیابی قرار می‌دهیم. سپس با توجه به نتایج به دست آمده، تلاش می‌کنیم مدل‌ها را بهینه‌سازی کرده و کارایی آن‌ها را افزایش دهیم. این فصل به عنوان یک مرحله حیاتی در گام‌های پیشروی پروژه می‌تواند به ما کمک کند تا مدل‌های پیش‌بینی خود را بهبود بخشیم و در نهایت نتایج دقیق‌تری در پروژه حاصل کنیم. از متداول‌ترین روش‌ها برای بهینه‌سازی مدل‌ها، تنظیم پارامترها، انتخاب ویژگی‌ها، و استفاده از تکنیک‌های متنوعی از جمله افزایش حجم داده، استفاده از مدل‌های پیچیده‌تر، و تغییر الگوریتم‌های آموزش است. در این فصل، ما به دنبال بهترین راهبردها برای دست‌یابی به یک مدل پیش‌بینی عالی و دقیق هستیم تا در نهایت به نتایج کاربردی و مفیدی دست پیدا کنیم.

داده های به دست آمده از فصل قبل به صورت نمودارهای زیر است:




ماتریس های confusion




مقایسه متدهای مختلف

:Accuracy

در دیتاهای آموزشی ما، هیچ یک از مدل‌ها دقت بالایی بیش از ۷۳ درصد ندارند. در مقایسه با مدل‌های دیگر، XGBoost به عنوان بهترین مدل با دقت عالی درخشیده و بدترین عملکرد به مدل Naïve Bayes تعلق دارد.

:Precision

در دیتاهای آموزشی ما، هیچ یک از مدل‌ها دقت بالایی بیش از ۷۴ درصد ندارند. XGBoost نیز از نظر دقت برترین مدل است و دارای بدترین عملکرد به مدل CART تعلق دارد.

:Recall

در دیتاهای آموزشی ما، هیچ یک از مدل‌ها دقت بالایی بیش از ۷۲ درصد ندارند. XGBoost نیز از نظر recall بهترین عملکرد را ارائه می‌دهد و بدترین عملکرد متعلق به مدل Naïve Bayes است.

F1 Score:

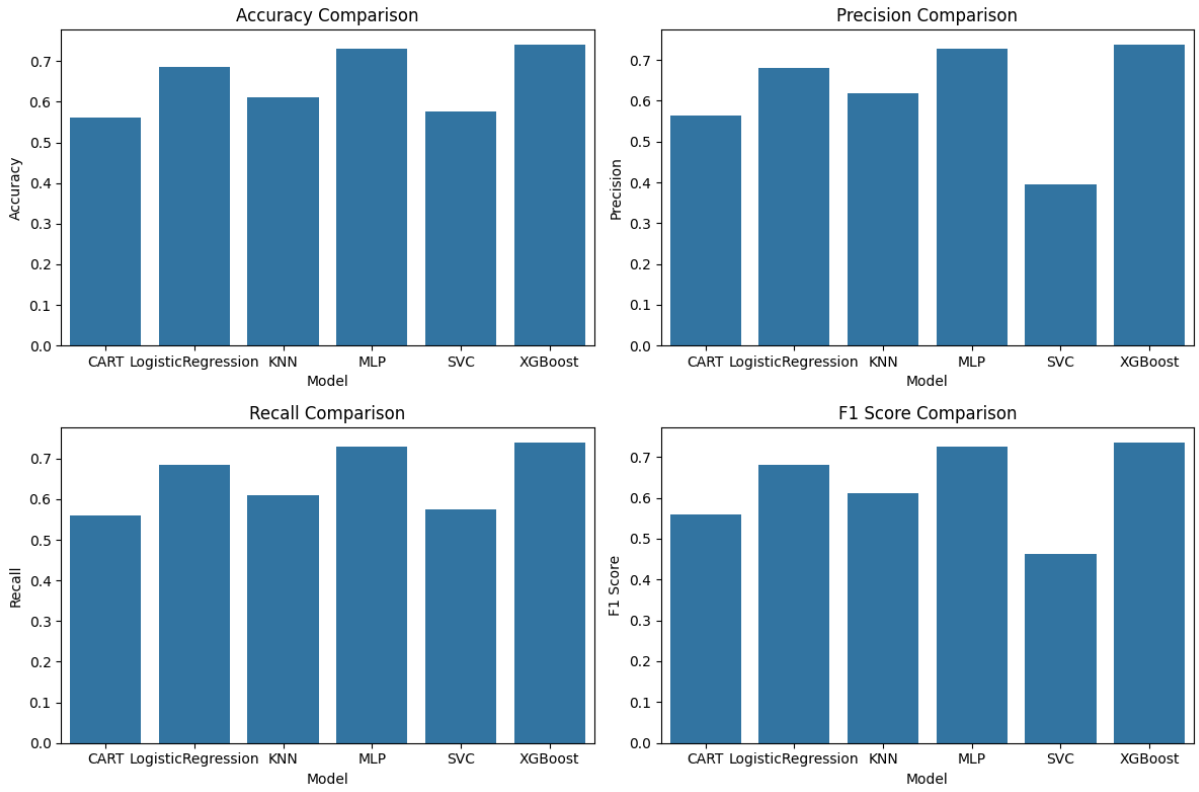
در دیتاهای آموزشی ما، هیچ یک از مدل‌ها F1 Score بالایی بیش از ۷۲ درصد ندارند. XGBoost به عنوان بهترین مدل از نظر F1 Score شناخته می‌شود و Naïve Bayes دارای بدترین عملکرد در این مورد است.

پیشرفت‌های به‌دست‌آمده در این پژوهش نشان می‌دهد که استفاده از مدل XGBoost به‌عنوان مدل اصلی تاثیر بسزایی در بهبود دقت و عملکرد کلی دارد. با اینکه مدل‌های دیگر نیز در مراحل مختلف مقایسه شدند، اما اینکه XGBoost به‌عنوان بهترین گزینه برجسته شود نشان‌دهنده قابلیت بالای این الگوریتم در مدیریت داده‌های پیچیده و گسترده می‌باشد.

بررسی معیارهای مختلف ارزیابی، از جمله دقت (Accuracy)، دقت مثبت (Precision)، بازخوانی (Recall) و اسکور F1 (F1 Score)، نشان می‌دهد که XGBoost به‌طور متوسط در هر یک از این معیارها بهترین عملکرد را از خود نشان می‌دهد. این تجزیه و تحلیل ارتقاء قابلیت پیش‌بینی و دقت در تصمیم‌گیری‌های آتی را از این مدل تایید می‌کند.

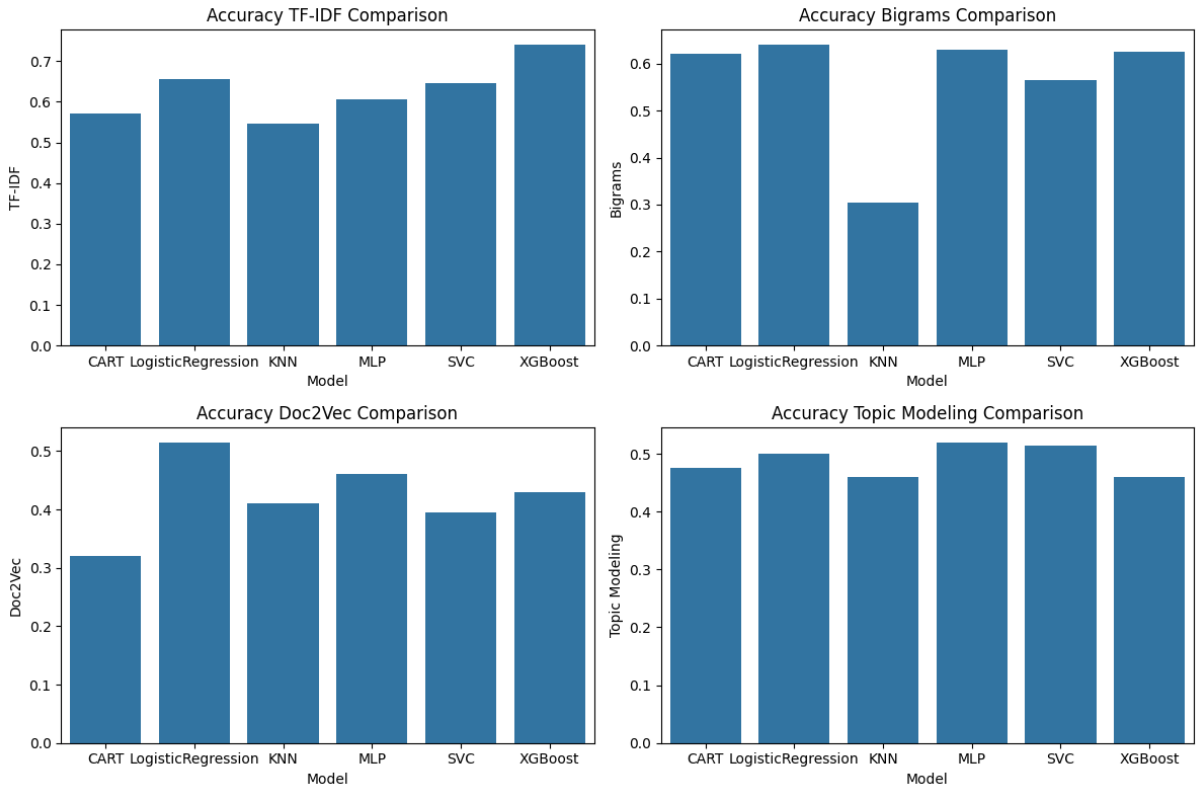
## مقایسه مدل ها با اعمال Optimization ها:

- مقایسه و بهینه‌سازی مدل‌ها با اعمال بهینه‌سازی Hstacking Text:




در این بررسی بهینه‌سازی، اعمال بهینه‌سازی Hstacking Text نتایج قابل توجهی در تاثیر مدل‌ها به‌ویژه بر دقت (accuracy) داشته است، که در تصویر نیز به وضوح مشهود است. تمام مدل‌ها به جز CART با افزایش دقت مواجه شده‌اند، به‌خصوص می‌توان تاثیر بسزایی را بر XGBoost مشاهده کرد. در حالی‌که این بهینه‌سازی بر دقت اثر مثبت گذاشته، تأثیر آن بر دقت تخصیصی (precision) تا حدی منفی بوده است، به‌جز برای مدل XGBoost که به نظر می‌رسد این بهینه‌سازی در آن تأثیر مناسبی نداشته باشد. همچنین، بر روی بازخوانی (recall) تأثیر زیادی نداشته و مدل‌ها تقریباً به همان ترتیب قبلی خود باقی‌مانده‌اند.در مورد امتیاز F1، بهینه‌سازی Hstacking Text تأثیر مثبتی داشته و موجب ارتقاء امتیاز F1 برای مدل‌های XGBoost و MLP شده است. این نتایج نشان‌دهنده این است که استفاده از بهینه‌سازی Hstacking Text می‌تواند بهبود قابل توجهی در عملکرد مدل‌ها، به‌ویژه در معیارهای دقت و امتیاز F1، ایجاد کند.

- مقایسه و بهینه‌سازی مدل‌ها با اعمال بهینه‌سازی Set Hyper Parameters:




مقایسه و بهینه‌سازی مدل‌ها با اعمال بهینه‌سازی تنظیم پارامترهای ابر:

در این تجزیه و تحلیل بهینه‌سازی، اعمال بهینه‌سازی تنظیم پارامترهای ابر نتایج چشم‌گیری در دقت مدل‌ها، به‌ویژه در جنبه دقت (accuracy)، به‌همراه داشته است. در زمینه دقت، بهینه‌سازی با استفاده از Bigram به مدل‌ها کاهش قابل توجهی داده است. این به‌ویژه در تصویر مشهود است که Bigram توانسته است دقت مدل‌ها را به حداقل برساند. از سوی دیگر، در استفاده از مدل Doc2Vec برای Logistic Regression، تأثیر منفی مشهود بوده و دقت این مدل را افزایش داده است.

در تحلیل دیگر مقایسه‌ها نیز اثرات متفاوتی مشاهده می‌شود که در تصویر به خوبی قابل مشاهده هستند. این تغییرات نشان‌دهنده این است که اعمال بهینه‌سازی تنظیم پارامترهای ابر به مدل‌ها به تعادل و بهبود در عملکرد آن‌ها منجر شده و این امر به تناسب با نوع مدل و استفاده از ویژگی‌های مختلف، نتایج متنوعی را به دنبال دارد.

از نظر زمانی، در اجرای کدها، مشاهده شد که بیشترین زمان مربوط به اجرای مدل MLP در هر دو حالت بهینه‌سازی بوده است. به وضوح مشاهده شد که عملیات اجرای MLP به علت پیچیدگی بالا و نیاز به محاسبات متعدد، زمان اجرای بیشتری را اشغال می‌کند.

علاوه بر این، در ترتیب زمانی دیگر نیز مشاهده شد که پس از MLP، مدل KNN و سپس SVC زمان کمتری برای اجرا به‌خود اختصاص داده‌اند. این نتایج نشان‌دهنده تفاوت‌های قابل توجه در زمان اجرا بین مدل‌هاست، که از اهمیت آن در انتخاب و استفاده از مدل‌های مختلف در محاسبات زمان‌بر برنامه‌ها و پروژه‌های مختلف خبر می‌دهد.

به‌طور کلی، در نظر گرفتن نتایج زمانی می‌تواند در انتخاب مدل مناسب بر اساس نیازهای پروژه و محدودیت‌های زمانی کمک مؤثری کند.

**منابع:**


1. https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

2. https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/