مسئله‌ای که من به عنوان پروژه نهایی درس روی آن کار کردم تحلیل اشعار فارسی و تشخیص شاعر آن‌ها بود و آزمایش‌هایی در این راستا انجام دادم از سه جهت قابل تقسیم به چند دسته است:

- مدل‌هایی که از آن‌ها استفاده کردم
  - روش‌های کلاسیک مانند SGD یا SVM یا Random Forest
  - Fasttext
  - LSTM
- واحدی که به عنوان ورودی مدل درنظر گرفتم
  - مصراع
  - قطعه شعر کامل
  - چند بیت
- متوازن بودن مجموعه داده
  - نامتوازن(imbalance)
  - متوازن(balance)

توضیحات مجموعه داده

مجموعه داده‌ای که از آن استفاده کردم مجموعه اشعار فارسی جمع شده از سایت گنجور بود که شامل ۱.۸ میلیون مصراع شعر از ۹۷ شاعر زبان و ادب فارسی است که البته اگر این شعرا را بر اساس تعداد مصراع‌هایی در این مجموعه داده دارند مرتب کنیم به ترتیب زیر می‌رسیم که متعلق به چند شاعر اول این ترتیب است

جامی: ۳۳۴۳۰	فردوسی: ۱۰۳۰۶۴	حافظ: ۹۱۵۴۵۶
سعدی: ۳۲۹۰۰	نظامی: ۶۷۱۱۱	مولوی: ۱۲۴۰۸۹
رشیدالدین میبدی: ۳۲۱۵۷	سنایی: ۵۴۸۷۱	عطار: ۱۲۳۵۱۱

همین‌طور که مشخص است این مجموعه بسیار نامتوازن است و به عنوان مثال حافظ که دارای بیشترین تعداد مصراع شعر در این مجموعه داده است به نسبت شاعر بعدی یعنی مولوی حدود ۸ برابر داده دارد و از ۱.۸ میلیون مصراع شعر از ۹۷ شاعر بیشتر از ۱.۳ میلیون شعر به همین چند شاعر اول اختصاص دارد و به خاطر همین میزان نامتوازن بودن مجموعه داده در گام اول برای آزمایش‌های متفاوت وابسته به نوع ورودی مدل‌ها تمرکز را روی چند شاعر اول بر اساس تعداد داده‌هایی که داشتند گذاشتم و سعی کردم مسئله را برای این شعرا حل کنم

آزمایش‌های مبتنی بر مصراع‌ها

تعداد شعرایی که برای این آزمایش‌ها انتخاب کردم ۶تا شاعر اول از روی همان جدول بالا بود چون یک اختلاف معنی‌دار به وجود می‌آمد و شعرا پایین‌تر به نسبت کلاس حافظ داده کمی داشتند و بعد از این انتخاب آزمایش‌های زیر صورت گرفت

- مدل SGD روی مجموعه داده نامتوازن: این آزمایش روی بردارهای ساخته شده بر مبنای tf-idf و مدل Stochastic Gradient Descent انجام گرفت و نتایج آن به صورت زیر بود

	precision	recall	f1-score	support
حافظ	83	83	83	183649
سنایی	23	15	18	11068
مولوی	43	37	40	24759
فردوسی	56	81	66	20386
عطار	45	39	42	24549
نظامی	31	34	32	13209
accuracy			70	277620
macro avg	47	48	47	277620
weighted avg	69	70	69	277620

همین‌طور که مشخص است دقت این آزمایش به ۷۰ درصد رسیده است اما این عدد قابل اتکا نیست چون میانگین f1-score کلاس‌ها به ۴۷ درصد رسیده است و یا تعدادی از کلاس‌ها وجود دارند که f1-score خیلی پایینی گرفته‌اند و به نوعی مدل نتوانسته به درستی این کلاس‌ها یاد بگیرد و اگر confusion matrix این مدل را نگاه کنید خیلی از داده‌های کلاس‌های دیگر را به اشتباه برچسب حافظ می‌زند اما دلیل بالا بودن معیار دقت به خاطر تعداد داده زیاد کلاس حافظ است و چون توانسته در آن کلاس به f1-score حدود ۸۳ درصد برسد معیار دقت را هم بالا کشیده است اما از یک دید دیگر هم می‌توان ارزیابی کرد که دقت ۷۰ درصد قابل اتکا نیست چون فرض کنید مدلی داشته باشیم که فارغ از ورودی‌ای که به آن می‌دهیم حافظ تشخیص بدهد و با توجه به تعداد داده‌هایی که داریم این مدل هم به دقت حدود ۶۶ درصد می‌رسید که دقت ۷۰ درصد که این مدل به آن رسیده اختلاف چندانی ندارد

(ناگفته نماند که جز آزمایش این مدل روی بردارهای tf-idf از fasttext هم برای embedding و بعد استفاده از همین مدل SGD استفاده کردم که نتیجه قابل قبولی نداشت و مدل SGD هم به چند دلیل از بین روش‌های کلاسیک انتخاب کردم که هم زمان آموزش کوتاه‌تری داشت و هم روی آزمایش‌هایی که انجام دادم معمولا نتیجه بهتری را داشت)

- مدل Fasttext روی مجموعه داده نامتوازن: این آزمایش روی بردارهای ساخته شده توسط embedding خود fasttext و با استفاده از مدل خود fasttext انجام شد و به نتایج زیر رسید

	precision	recall	f1-score	support
حافظ	80	93	86	183649
سنایی	39	08	13	11068
مولوی	54	38	44	24759
فردوسی	77	76	77	20386
عطار	54	38	44	24549
نظامی	49	28	36	13209
accuracy			76	277620
macro avg	59	47	50	277620
weighted avg	72	76	73	277620

نتیجه‌ای که روی این مدل و آزمایش به آن رسیدیم به نسبت آزمایش قبلی مطلوب‌تر هست چون هم به صورت کلی دقت بالاتری دارد و هم میانگین f1-score کلاس‌ها مقدار بالاتری گرفته است اما همچنان هم میانگین f1-score کلاس‌ها چندان بالا نیست و بعضی از کلاس‌ها f1-score خیلی پایینی گرفته‌اند و همچنان هم اگر confusion matrix این آزمایش را ببینید در خیلی از موارد مدل داده‌ها را به اشتباه برچسب حافظ می‌زند

یک نتیجه کلی که می‌توان بر مبنای آزمایش‌های مبتنی بر مصراع‌ها گرفت این هست که مصراع به عنوان واحد ورودی و تصمیم‌گیری مدل چندان اطلاعات در خود ندارد چون خیلی کوتاه است و ویژگی‌هایی از شعر مثل قالب شعری یا استفاده از کلمات خاص مثل تخلص شعرا را نمی‌تواند به خوبی حفظ کند و در گام

بعدی به جای درنظر گرفتن مصراع‌ها به عنوان واحد ورودی و تصمیم‌گیری به سراغ قطعه شعرهای کامل رفتم و با استفاده از فیلدهایی در مجموعه داده وجود داشت مصراع‌ها بهم متصل کردم و از روی آن‌ها شعرها را ساختم و بعد از این کار توزیع داده‌ها در کلاس‌ها به صورت زیر بود

سنایی: ۱۱۹۶	مولوی: ۱۶۱۰	حافظ: ۴۵۳۸۹
اقبال لاهوری: ۹۷۲	سعدی: ۱۴۸۸	عطار: ۳۹۳۸
انوری: ۹۰۱	رشیدالدین میبدی: ۱۳۳۷	صائب تبریزی: ۲۴۸۸

جدول بالا شامل کلاس‌ها با بیشترین تعداد قطعه شعر کامل در مجموعه داده است و همین‌طور هم که مشخص است همچنان میزان بالا نامتوازنی داده‌ها هم پا بر جا است و شبیه به حالت آزمایش‌های روی مصراع‌ها سعی کردم در گام اول مسئله را روی ۷ کلاس حل کنم چون بعد از آن یک اختلاف نسبی رخ می‌دهد

آزمایش‌های مبتنی بر قطعه شعرهای کامل

- مدل SGD روی مجموعه داده نامتوازن: این آزمایش را روی بردارهای ساخته شده بر مبنای tf-idf و مدل Stochastic Gradient Descent انجام دادم و حاصل آن نتایج زیر بود

	precision	recall	f1-score	support
حافظ	94	92	93	9042
رشیدالدین میبدی	100	100	100	283
سعدی	45	54	49	280
سنایی	50	52	51	257
صائب تبریزی	55	57	56	518
عطار	68	76	72	783
مولوی	82	84	83	326
accuracy			87	11489
macro avg	70	73	72	11489

weighted avg	88	87	87	11489

همین‌طور که می‌توان دید دقت در کنار مقدار میانگین f1-score کلاس‌ها به صورت چشم‌گیری رشد داشته است ولی تا حدی این رفتار قابل انتظار بود چون قطعه شعر کامل به میزان خوبی بلندتر از مصراع‌ها است و اطلاعاتی که در اختیار مدل قرار می‌گیرد به نسبت قبل بیشتر از است و مدل بهتر می‌تواند تصمیم‌گیری کند و علاوه بر این‌ها مشکل خیلی پایین بودن f1-score بعضی از کلاس‌ها کم‌رنگ‌تر شده است و همه کلاس‌ها حدودا به f1-score حداقل ۵۰ درصد رسیده‌اند اما همچنان بخش عمده‌ای از اشتباهات مدل بین کلاس حافظ و دیگر کلاس‌ها است که از روی confusion matrix می‌توان آن را مشاهده کرد

- مدل Fasttext روی مجموعه داده نامتوازن: این آزمایش روی بردارهای ساخته شده توسط embedding خود fasttext و با استفاده از مدل خود fasttext انجام شد و به نتایج زیر رسید

	precision	recall	f1-score	support
حافظ	90	98	94	9042
رشیدالدین میبدی	95	96	96	283
سعدی	35	09	15	280
سنایی	00	00	00	257
صائب تبریزی	80	66	72	518
عطار	72	58	64	783
مولوی	70	48	57	326
accuracy			88	11489
macro avg	63	54	57	11489
weighted avg	84	88	86	11489

این مدل توانسته به نسبت آزمایش قبل دقت کلی بهتری را بگیرد اما به صورت چشم‌گیری میانگین f1-score کلاس‌ها کاهش پیدا کرده است و دوباره شاهد بروز f1-score خیلی کم یا حتی نزدیک به صفر در بعضی از کلاس‌ها باشیم و تنها نتیجه اندکی بهتری که روی کلاس حافظ گرفتیم باعث شده

تا دقت کلی افزایش پیدا کند و همانند قبل همچنان مدل در بسیار از موارد داده‌ها را به اشتباه برچسب حافظ می‌زند که روی confusion matrix قابل مشاهده است

- مدل LSTM روی مجموعه داده نامتوازن: در این آزمایش برای ساختن بردارها از embedding از پیش آموزش داده شده fasttext استفاده کردم و ساختار مدلی هم که داشتم به ترتیب شامل همین لایه embedding و یک تک لایه LSTM با ۶۴تا hidden state و یک لایه linear و یک لایه softmax بود اما متاسفانه آزمایش موفقیت آمیزی نبود و در حین آموزش تنها در epoch اول مقدار تابع هزینه مدل کاهش پیدا می‌کرد که آن هم بسیار اندک بود و در epochهای بعدی هم مقدار تابع هزینه کاملا ثابت بود و تغییری نمی‌کرد و دلیلش هم به خاطر کم بودن تعداد داده برای آموزش در کلاس‌های جز حافظ بود که در کل می‌توان گفت در میان شاعرهای انتخابی حدود ۶۰ هزار قطعه شعر کامل وجود داشت که اگر آن را به صورت ۸۰ ۲۰ هم تقسیم کنیم برای فرایند آموزش حدود ۴۵ هزار داده خواهیم داشت که مجموع سهم همه کلاس‌ها به جز حافظ حدود ۱۰ هزار داده می‌شود و بعضی کلاس‌ها حدود هزار داده برای آموزش دارند

با توجه به مشکل ذکر شده راجع به کم بودن تعداد داده باید واحد ورودی را به نحوی انتخاب می‌کردیم که علاوه بر اینکه یک میزان قابل قبولی اطلاعات در خودش دارد اما چندان بلند نباشد که باعث شود تعداد داده‌ها خیلی کم شوند و این واحد میانی به صورت کلی می‌تواند چند بیت از شعرها باشد که با توجه به آزمایش‌هایی که انجام دادم چهار بیت می‌تواند واحد به نسبت مناسبی برای هر دو چالش باشد و با این تصمیم توزیع داده‌ها روی چند کلاس با بیشترین تعداد داده به شکل زیر می‌شود

سنایی: ۷۰۰۷	فردوسی: ۱۳۱۴۳	حافظ: ۱۳۲۸۶۷
سعدی: ۴۸۲۱	نظامی: ۸۵۶۲	عطار: ۱۷۹۹۹
احمد شاملو: ۴۴۵۲	رشیدالدین میبدی: ۷۵۴۳	مولوی: ۱۶۰۶۲

با توجه به توضیحات قبل باز هم در گام اول سعی کردم مسئله را برای ۷تا شاعر اول بر اساس تعداد داده‌ها حل کنم چون اختلاف نسبی وجود دارد و در مجموع برای این ۷ شاعر می‌توان دید که به جای حدود ۶۰ هزار داده به حدود ۲۰۰ هزار داده رسیدیم که می‌توان گفت تا حدی چالشی که سر کم بودن تعداد داده‌ها داشتیم حل شده است

آزمایش‌های مبتنی برای چهار بیت:

- مدل SGD روی مجموعه داده نامتوازن: این آزمایش روی بردارهای ساخته شده بر مبنای tf-idf و مدل Stochastic Gradient Descent انجام گرفت و نتایج آن به صورت زیر بود

	precision	recall	f1-score	support
حافظ	93	91	92	26512
رشیدالدین میبدی	98	98	98	1574
سنایی	44	44	44	1461
عطار	72	73	72	3624
فردوسی	93	94	94	2613
مولوی	69	71	70	3236
نظامی	66	81	73	1616
accuracy			86	40636
macro avg	76	79	78	40636
weighted avg	87	86	86	40636

نتیجه این مدل از چند نظر قابل اهمیت است چون علاوه بر اینکه از نظر دقت کلی به مقدار قابل قبولی رسیده است و جزو چند نتیجه اول به حساب می‌آید به بهترین میانگین f1-score به نسبت تمامی آزمایش‌های قبل شده است و جز یک کلاس f1-score کلاس‌ها به بالای ۷۰ درصد رسیده است و آن کلاس استثنا یعنی سنایی هم به نسبت بهترین f1-score که در نتایج قبل گرفته است کاهش چشم‌گیری نداشته است و این نشان از حدس درستی که راجع به واحد میانی بین مصراع‌ها و قطعه شعر کامل است اما همچنان اشتباهات عمده مدل برای برچسب حافظ زدن به کلاس‌های دیگر است که اصلی‌ترین دلیل آن هم تعداد داده بسیار بالاتر به نسبت بقیه کلاس‌ها است

- مدل Fasttext روی مجموعه داده نامتوازن: این آزمایش روی بردارهای ساخته شده توسط embedding خود fasttext و با استفاده از مدل خود fasttext انجام شد و به نتایج زیر رسید

	precision	recall	f1-score	support

حافظ	90	96	93	26512
رشیدالدین میبدی	99	98	98	1574
سنایی	61	23	34	1461
عطار	77	75	76	3624
فردوسی	96	94	95	2613
مولوی	77	66	71	3236
نظامی	80	73	77	1616
accuracy			88	40636
macro avg	83	75	78	40636
weighted avg	87	88	87	40636

نتیجه این مدل هم همانند آزمایش قبلی بسیار مطلوب است و حتی روی معیار دقت ۲ درصد دقت بهتری را کسب کرده است اما میانگین f1-score آن با آزمایش قبلی یکسان است و این بهبود دقت با احتساب ثابت بودن میانگین f1-score اصلی‌ترین دلیلش بهبودی اندکی هست که این مدل روی کلاس حافظ به نسبت آزمایش قبلی گرفته است ولی معتقدم آزمایش قبلی نتیجه بهتر و قابل اتکاتری دارد و یکی از دلایل آن کاهش ۱۰ درصدی مقدار f1-score روی کلاس سنایی است و دلیل دیگری هم که دارم کاهش ۴ درصدی میانگین recall کلاس‌ها در این مدل به نسبت آزمایش قبلی است

- مدل SGD روی مجموعه داده متوازن: این آزمایش را روی مجموعه داده متوازن به این معنی که تعداد داده‌های هر کلاس تقریبا مساوی با دیگر کلاس‌ها باشد و روی بردارهای ساخته شده بر مبنای tf-idf و مدل Stochastic Gradient Descent انجام دادم و نتایج آن به صورت زیر بود

	precision	recall	f1-score	support
حافظ	81	78	80	3898
رشیدالدین میبدی	98	99	98	1511

سنایی	63	63	63	1426
عطار	75	74	74	1800
فردوسی	95	95	95	1709
مولوی	71	72	72	1718
نظامی	81	87	84	1620
accuracy			81	13682
macro avg	81	81	81	13682
weighted avg	81	81	81	13682

این مدل به نسبت تمامی آزمایش‌های قبلی میانگین recall و f1-score بهتر گرفته است اما دقت آن کاهش حدود ۷ درصدی داشته است که به خاطر متوازن شدن داده و نتیجه نه چندان بالا اما معقول در کلاس حافظ است و f1-score کلاس‌ها حتی در کلاس سنایی همگی به بالای ۶۰ درصد رسیده است و اگر confusion matrix این مدل را هم نگاه کنیم متوجه می‌شیم چالشی که سر اشتباه برچسب حافظ زدن روی آزمایش‌های قبلی تا حدی چشم‌گیری کاهش پیدا کرده است و درنهایت وابسته به اهمیت و نیاز می‌توان تصمیم گرفت که بالاتر بودن ۷ درصدی دقت کارایی بیشتری دارد یا بالاتر بودن recall و f1-score کلاس‌ها

- مدل Fasttext روی مجموعه داده نامتوازن: این آزمایش را روی مجموعه داده متوازن به این معنی که تعداد داده‌های هر کلاس تقریبا مساوی با دیگر کلاس‌ها باشد و روی بردارهای ساخته شده توسط embedding خود fasttext و با استفاده از مدل خود fasttext انجام دادم و به نتایج زیر رسید

	precision	recall	f1-score	support
حافظ	75	86	80	3898
رشیدالدین میبدی	98	98	98	1511
سنایی	66	48	56	1426
عطار	75	72	73	1800

فردوسی	95	93	94	1709
مولوی	72	69	70	1718
نظامی	83	82	82	1620
accuracy			80	13682
macro avg	80	78	79	13682
weighted avg	80	80	79	13682

برای این مدل به صورت کلی می‌توان گفت تا حدی ویژگی‌های مثبتی که در آزمایش قبل به آن‌ها اشاره کردیم را دارا هست اما همچنان نتایج آزمایش قبلی از نظر معیارهای دقت یا recall یا f1-score عملکرد بهتری را دارد

- مدل LSTM روی مجموعه داده متوازن: در این آزمایش را روی مجموعه داده متوازن به این معنی که تعداد داده‌های هر کلاس تقریبا مساوی با دیگر کلاس‌ها باشد و برای ساختن بردارها از embedding از پیش آموزش داده شده fasttext استفاده کردم و ساختار مدلی هم که داشتم به ترتیب شامل همین لایه embedding و یک تک لایه LSTM با ۶۴تا hidden state و یک لایه linear و یک لایه softmax بود و این مدل را روی ۳۰تا epoch آموزش دادم و به نتایج زیر رسید

	precision	recall	f1-score	support
حافظ	71	86	78	3898
رشیدالدین میبدی	96	98	97	1511
سنایی	46	35	40	1426
عطار	60	58	59	1800
فردوسی	93	87	90	1709
مولوی	62	54	58	1718
نظامی	71	65	68	1620
accuracy			72	13682

macro avg	71	69	70	13682
weighted avg	71	65	71	13682

توضیحات کد:

- پوشه data: در این پوشه فایل‌های مشترک بین تمامی آزمایش‌ها مانند مجموعه داده اصلی یا مجموعه داده تمیز و آماده شده برای فرایندهای آموزش و ارزیابی و چنین فایل‌هایی قرار دارد
- پوشه linear
  - فایل classifier.py: این فایل شامل کدها مرتبط با آموزش و ارزیابی مدل‌های کلاسیک است
  - پوشه data: فایل مدل‌های کلاسیک بعد از آموزش و ارزیابی در این پوشه ذخیره می‌شود
- پوشه fasttext
  - فایل dataset.py: این فایل شامل کدهایی است که از روی مجموعه داده آماده و تمیز شده مجموعه داده با ساختار متناسب برای آموزش و ارزیابی مدل fasttext را فراهم کند
  - فایل classifier.py: این فایل با استفاده از داده‌های فراهم شده مدل fasttext را آموزش و ارزیابی می‌کند
  - پوشه data: در این پوشه مجموعه داده‌های متناسب با ساختار مدل fasttext ذخیره می‌شود
- پوشه lstm
  - فایل classifier.py: این فایل شامل کدهایی برای تعریف و ساختار شبکه و همچنین آموزش شبکه و ذخیره checkpointها است
  - فایل evaluate.py: این فایل شامل کدهای مرتبط با ارزیابی مدل آموزش دیده شده است
  - پوشه data: این پوشه شامل checkpointهای ذخیره در حین آموزش است
- پوشه results: این پوشه شامل ارزیابی‌های آزمایش‌های انجام شده است
- فایل app.py: این فایل دمو وبی نوشته شده با استفاده از کتابخانه streamlit است و نحوه اجرا آن به این صورت است: `streamlit run app.py`
- فایل dataset.py: این فایل شامل کدهایی برای تمیز و آماده‌سازی مجموعه داده اولیه برای استفاده در فرایند آموزش و ارزیابی است که شامل چندین تابع برای ساختن مجموعه داده‌هایی با واحد(مصراع/شعر کامل/چند بیت) متفاوت است

- فایل data_balancer.py: این فایل شامل کدهایی برای متوازن کردن مجموعه داده بر اساس تعداد داده‌های هر کلاس است
- فایل requirements.txt: شامل لیست کتابخانه‌هایی است که در کدها از آن‌ها استفاده شده است