۱. مجموعه داده

مجموعه داده نامتعادل استفاده شده vehicle0 نام‌دارد. این مجموعه شامل ۸۴۶ نمونه است که ۶۴۵ عدد به کلاس منفی (۰) و بقیه به کلاس مثبت تعلق دارند. هر نمونه دارای ۱۸ ویژگی با مقادیر صحیح است . شکل زیر مختصری از ویژگی ها و محدوده آنها را نشان می دهد.

Attribute	Domain	Attribute	Domain
Compactness	[73, 119]	Length_rectangular	[118, 188]
Circularity	[33, 59]	Major_variance	[130, 320]
Distance_circularity	[40, 112]	Minor_variance	[184, 1018]
Radius_ratio	[104, 333]	Gyration_radius	[109, 268]
Praxis_aspect_ratio	[47, 138]	Major_skewness	[59, 135]
Max_length_aspect_ratio	[2, 55]	Minor_skewness	[0, 22]
Scatter_ratio	[112, 265]	Minor_kurtosis	[0, 41]
Elongatedness	[26, 61]	Major_kurtosis	[176, 206]
Praxis_rectangular	[17, 29]	Hollows_ratio	[181, 211]
Class	{positive,negative}		

۲. خلاصه ای از الگوریتم های Boosting پیاده سازی شده + جزییات پیاده سازی

۱.۲ AdaBoost M2

این الگوریتم نسخه بهبود یافته AdaBoostM1 است و معیار خطای جدیدی به نام Pseudo-loss را تعریف می کند. این معیار خطا میزان اعتماد طبقه بند ضعیف به پیش بینی را نیز مد نظر قرار می دهد و این مقدار اعتماد لزومی ندارد صرفا از جنس احتمال باشد.

$$\epsilon_t = \frac{1}{2} \sum_{(i,y)\in B} D_t(i,y)\big(1 - h_t(x_i, y_i) + h_t(x_i, y)\big).$$

توضیحات نحوه عملکرد و مراحل پیاده سازی: کلیت این روش به این صورت است که به هر نمونه‌در مجموعه داده یک وزن اختصاص داده می شود این وزن ها در ابتدا به صورت Uniform مقدار دهی می شوند. سپس **در K مرحله الگوریتم زیر اجرا می شود:**

۱. آموزش طبقه بند ضعیف k ام روی داده ها با وزن های مشخص مرحله قبل
۲. محاسبه Pseudo-loss
۳. محاسبه beta=loss/1-loss مخصوص طبقه بند این مرحله
۴. به روز رسانی وزن نمونه ها با توجه به beta و فرمول زیر به همراه نرمالیزه سازی وزن های جدید:

$$D_{t+1}(i,y) = \frac{D_t(i,y)}{Z_t} \cdot \beta_t^{(1/2)(1+h_t(x_i,y_i)-h_t(x_i,y))}$$

۵. در صورتی که k به آخر نرسیده حلقه تکرار می شود.
۶.برای پیشبینی کلاس نمونه جدید (تست) : به ازای هر کلاس مجموع حاصلضرب وزن مربوط به هر طبقه بند در میزان اعتماد طبقه بند در تعلق نمونه به آن کلاس مشخص بدست آمده و کلاسی که بیشترین امتیاز را دارد انتخاب می شود.

- **طبقه بند ضعیف در تمام الگوریتم ها درخت تصمیم با حداکثر عمق ۲ در نظر گرفته شده است. برای استفاده از الگوریتم درخت تصمیم از کتابخانه SK-Learn استفاده شده است که از الگوریتم CART استفاده کرده و قابلیت برگرداندن احتمال تعلق به کلاس ها را نیز دارا است.**

SMOTE Boost .۲.۲

این روش برای Boosting از الگوریتم AdaBoostM2 استفاده می کند. اما نکته ای که آن را از دیگر الگوریتم ها متمایز می کند عمل OverSampling است که روی داده های کلاس Minority انجام می شود و مشکل نامتعادل بودن مجموعه داده را حل می کند. OverSampling به این صورت انجام می شود که ابتدا یک مدل KNN روی داده های کلاس اقلیت آموزش داده می شود. سپس به صورت تصادفی (به تعداد نیاز برای برقراری تعادل) از داده های کلاس اقلیت انتخاب می شود حال برای هر کدام از این داده ها از بین k نزدیک ترین همسایه آن یکی به صورت تصادفی انتخاب و داده ساختگی جدید با توجه به اختلاف دو بردار همسایه ایجاد می شود.

مراحل پیاده سازی :

۱. وزن دهی Uniform به نمونه های محموعه داده
۲. تشخیص کلاس minority و برازش مدل knn به نمونه های آن
۳. برای k مرحله قرم های زیر تکرار می شود:
  ۱.۳. انجام Over Sampling به روش Knn و وزن دهی آنها به صورت uniform
  ۲.۳. آموزش طبقه بند ضعیف
  ۳.۳. محاسبه Pseudo-loss
  ۴.۳. بروز رسانی وزن داده های اصلی و نرمالیزه کردن آنها (مشابه با آنچه قبلا توضیح داده شد.)
۴. برای پیش بینی نمونه جدید نیز مثل قبل عمل شده و کلاسی که بیشترین score را بر اساس معادله زیر داشته باشد انتخاب می شود.

$$h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(x, y).$$

RUSBoost .۳.۲

این روش از Under Sampling داده های کلاس اکثریت برای حفظ تعادل مجموعه داده استفاده می کند. به اینصورت که به شکل تصادفی داده های کلاس اکثریت را حذف می کند تا به تعادل مورد نظر بین داده های دو کلاس برسد.

مراحل پیاده سازی :

۱. وزن دهی Uniform به نمونه های محموعه داده
۲. تشخیص کلاس Majority
۳. برای k مرحله قرم های زیر تکرار می شود:
  ۱.۳. انجام Under Sampling به روش توضیح داده شده
  ۲.۳. آموزش طبقه بند ضعیف
  ۳.۳. محاسبه Pseudo-loss
  ۴.۳. بروز رسانی وزن داده های اصلی و نرمالیزه کردن آنها (مشابه با آنچه قبلا توضیح داده شد.)
۴. برای پیش بینی نمونه جدید نیز مثل قبل عمل شده و کلاسی که بیشترین score را داشته باشد انتخاب می شود.

Random Balance Boost .۴.۲

این روش ترکیبی از دو ایده SMOTE و RUS می باشد. الگوریتم Boosting همان AdaBoostM2 است. برای آموزش هر طبقه بند ضعیف یک مجموعه داده طبق الگوریتم زیر ساخته می شود:

۱. به صورت تصادفی سایز کلاس اکثریت مشخص می شود. (این عدد بایستی بین ۲ تا حداکثر تعداد داده های کلاس اکثریت باشد.)
۲. با توجه به سایز کلاس اکثریت سایز جدید کلاس اقلیت نیز مشخص می شود.

$$\text{Total\_size} - \text{new majority size} \quad (1)$$

۳. دو حالت کلی داریم :
۱. سایز کلاس **اکثریت جدید کمتر** از سایز کلاس **اکثریت اصلی** است:
- در این حالت بایستی به صورت تصادفی نمونه هایی از کلاس اکثریت کنارگذاشته شوند.
- با توجه به رابطه ۱ تعداد نمونه های کلاس اقلیت بایستی بیشتر از تعداد نمونه های آن در مجموعه داده اصلی شود لذا با روش knn که قبلا توضیح داده شده این داده های ساختگی را برای رسیدن به سایز مشخص شده ایجاد می کنیم.

۲. سایز کلاس **اکثریت** جدید **بیشتر** از سایز **کلاس اکثریت اصلی** است:
- در این حالت بایستی با روش knn که قبلا توضیح داده شده داده های ساختگی را برای کلاس اکثریت جهت رسیدن به سایز مشخص شده ایجاد می کنیم.
- با توجه به رابطه ۱ تعداد نمونه های کلاس اقلیت بایستی کمتر از تعداد نمونه های آن در مجموعه داده اصلی شود لذا بایستی به صورت تصادفی نمونه هایی از کلاس اقلیت کنارگذاشته شوند.

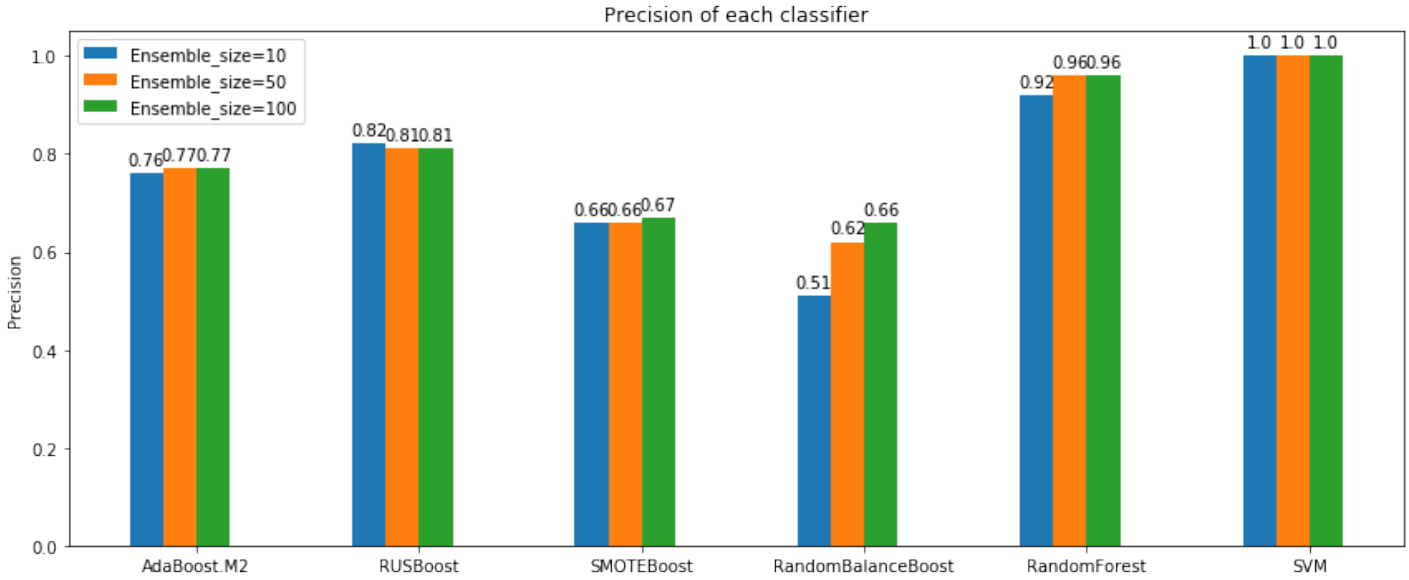
حال که **الگوریتم Random Balance** به همراه جزییات پیاده سازی آن توضیح داده شد **مراحل پیاده سازی الگوریتم کلی** به شرح زیر است :

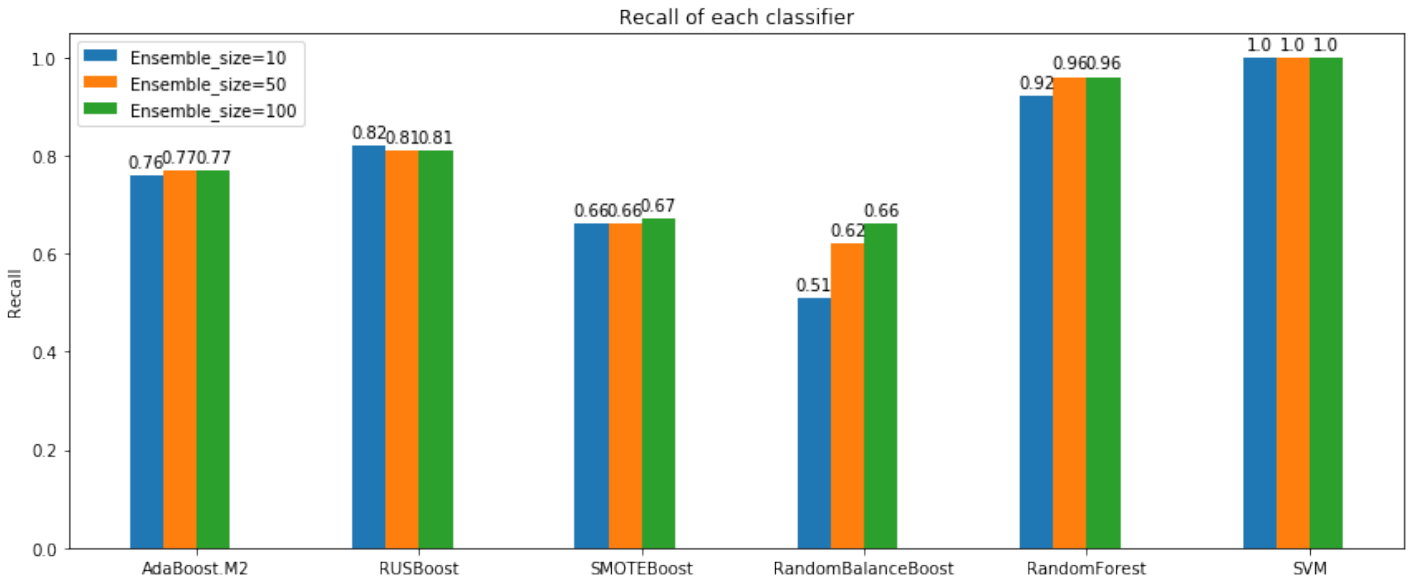
۱. وزن دهی Uniform به نمونه های محموعه داده
۲. برای k مرحله قرم های زیر تکرار می شود:
   ۱.۳. انتخاب سایز جدید برای کلاس اکثریت و اقلیت (تصادفی)
   ۲.۳. اعمال الگوریتم Random Balance برای تولید داده آموزشی (توضیح داده شده)
   * **نکته** : طبق گفته مقاله وزن مربوط به داده های ساختگی برابر ورن اولیه در نظر گرفته شده برای داده هاست یعنی اگر m نمونه داشته باشیم وزن هر داده جدید ۱ تقسیم بر m است.
   ۳.۳. آموزش طبقه بند ضعیف
   ۴.۳. محاسبه Pseudo-loss
   ۵.۳. بروز رسانی وزن داده های اصلی و نرمالیزه کردن آنها (مشابه با آنچه قبلا توضیح داده شد.)

۳. برای پیش بینی نمونه جدید نیز مثل قبل عمل شده و کلاسی که بیشترین scoreرا داشته باشد انتخاب می شود.

۳. نتایج بدست آمده و تحلیل آنها

زمان آموزش RUSBOOST به مقدار قابل توجهی از روش SMOTE و Random Balance Boost بیشتر و دقت آن به مراتب بالاتر است. چرا که بخش ایجاد داده ساختگی را که بر پایه Knn است ندارد و فقط به صورت تصادفی تعدادی از داده های کلاس اکثریت را دور می ریزد.
جدول زیر نتایج Precison و Recall هر روش را به تفکیک اندازه مجمع نشان می دهد. طبق نمودار زیر بهترین عملکرد را SVM دارد پس از آن و از بین Ensemble method ها Random Forest بهترین عملکرد را دارد. از بین روش های Boosting پیاده سازی شده روش RUSBoost در جایگاه نخست قرار دارد پس از آن AdaBoostM2.






```
Average Classifier Precision for AdaBoost :          0.77
Average Classifier Precision for RUSBoost :          0.82
Average Classifier Precision for SMOTEBoost :        0.66
Average Classifier Precision for RandomBalanceBoost :0.6
Average Classifier Precision for RandomForest :      0.95
Average Classifier Precision for SVM :               1.0
---------------------------------------------------------------
*Best performing method based on Average Precision of classifiers: "SVM"

*Best Performing Ensemble Classifier is "Random Forset" Runner up (second best) is RUSBOOST

-------------------------------------------------------------------------------

Taha Samavati - 98722134 - Analysis of final results
```

۴. بخش امتیازی ANOVA Test

در این آزمون برآنیم تا تشخیص دهیم آیا میان داده های موجود در چند دسته از نظر آماری تفاوت چندانی وجود دارد یا خیر. این آزمون آنالیز واریانس نام دارد و برای محاسبه آن قدم های زیر بایستی طی شود که در کد نیز به همین صورت پیاده سازی شده است:

۱. ابتدا مجموعه داده ۱۰ بار و به صورت تصادفی به بخش های train و test تقسیم می شود.
۲. برای هر طبقه بند ۱۰ دقت بدست می آید و ۶ طبقه بند داریم.
۳. داده ما دارای ۶ دسته و هر دسته شامل ۱۰ داده است.
۴. ابتدا میانگین کل دقت ها را محاسبه می کنیم و آنرا overall_mean نامگذاری می کنیم.
۵. واریانس درون دسته ای را محاسبه می کنیم.
به ازای هر گروه (هر طبقه بند) واریانس داده هایش را حساب می کنیم.سپس نتایج بدست آمده را جمع می کنیم.
۶. واریانس بین دسته ای را حساب می کنیم.
به ازای هر گروه مجذور اختلاف میانگین داده های آن از میانگین کل را حساب می کنیم. سپس نتایج را با هم جمع می کنیم.
۷. میانگین مربعات درون دسته ای و میانگین مربعات بین دسته ای را از فرمول های زیر محاسبه می کنیم:

میانگین مربعات درون دسته ای = درجه آزادی (تعداد دسته ها – تعداد کل داده ها) / واریانس درون دسته ای
میانگین مربعات بین دسته ای = درجه آزادی (۱ – تعداد دسته ها) / واریانس بین دسته ای

۸. در نهایت معیار F را بدست می آوریم که از تقسیم میانگین مربعات بین دسته ای به درون دسته ای بدست می آید. که در اینجا برابر ۴.۹۰ است. مقدار p را نیز بدست می آوریم که کمتر از ۰.۰۵ است.

در پایان به جدول توزیع F به ازای $p<0.05$ مراجعه کرده و با توجه به درجه آزادی های بدست آمده (۵۴و ۵) مقدار F را بدست می آوریم. که در اینجا ۱.۹۳ بدست می آید. چون ۱.۹۳ کمتر از ۴.۹ است لذا می توان گفت **میان داده های دسته های مختلف که همان دقت طبقه بند های مختلف است از نظر آماری اختلاف معنا داری وجود دارد. در صفحه بعد جدول مربوط به توزیع F آورده شده است.**

یادداشت پایانی:

- نمودار های ROC همگی در داخل نوت بوک رسم شده اند.
- تمامی نتایج بدست آمده به همراه توضیحات در نوت بوک موجود است.

	DF1	α = 0.10																	
DF2	1	2	3	4	5	6	7	8	9	10	12	15	20	24	30	40	60	120	Inf
1	39.863	49.5	53.593	55.833	57.24	58.204	58.906	59.439	59.858	60.195	60.705	61.22	61.74	62.002	62.265	62.529	62.794	63.061	63.328
2	8.5263	9	9.1618	9.2434	9.2926	9.3255	9.3491	9.3668	9.3805	9.3916	9.4081	9.4247	9.4413	9.4496	9.4579	9.4662	9.4746	9.4829	9.4912
3	5.5383	5.4624	5.3908	5.3426	5.3092	5.2847	5.2662	5.2517	5.24	5.2304	5.2156	5.2003	5.1845	5.1764	5.1681	5.1597	5.1512	5.1425	5.1337
4	4.5448	4.3246	4.1909	4.1073	4.0506	4.0098	3.979	3.9549	3.9357	3.9199	3.8955	3.8704	3.8443	3.831	3.8174	3.8036	3.7896	3.7753	3.7607
5	4.0604	3.7797	3.6195	3.5202	3.453	3.4045	3.3679	3.3393	3.3163	3.2974	3.2682	3.238	3.2067	3.1905	3.1741	3.1573	3.1402	3.1228	3.105
6	3.776	3.4633	3.2888	3.1808	3.1075	3.0546	3.0145	2.983	2.9577	2.9369	2.9047	2.8712	2.8363	2.8183	2.8	2.7812	2.762	2.7423	2.7222
7	3.5894	3.2574	3.0741	2.9605	2.8833	2.8274	2.7849	2.7516	2.7247	2.7025	2.6681	2.6322	2.5947	2.5753	2.5555	2.5351	2.5142	2.4928	2.4708
8	3.4579	3.1131	2.9238	2.8064	2.7265	2.6683	2.6241	2.5894	2.5612	2.538	2.502	2.4642	2.4246	2.4041	2.383	2.3614	2.3391	2.3162	2.2926
9	3.3603	3.0065	2.8129	2.6927	2.6106	2.5509	2.5053	2.4694	2.4403	2.4163	2.3789	2.3396	2.2983	2.2768	2.2547	2.232	2.2085	2.1843	2.1592
10	3.285	2.9245	2.7277	2.6053	2.5216	2.4606	2.414	2.3772	2.3473	2.3226	2.2841	2.2435	2.2007	2.1784	2.1554	2.1317	2.1072	2.0818	2.0554
11	3.2252	2.8595	2.6602	2.5362	2.4512	2.3891	2.3416	2.304	2.2735	2.2482	2.2087	2.1671	2.1231	2.1	2.0762	2.0516	2.0261	1.9997	1.9721
12	3.1766	2.8068	2.6055	2.4801	2.394	2.331	2.2828	2.2446	2.2135	2.1878	2.1474	2.1049	2.0597	2.036	2.0115	1.9861	1.9597	1.9323	1.9036
13	3.1362	2.7632	2.5603	2.4337	2.3467	2.283	2.2341	2.1954	2.1638	2.1376	2.0966	2.0532	2.007	1.9827	1.9576	1.9315	1.9043	1.8759	1.8462
14	3.1022	2.7265	2.5222	2.3947	2.3069	2.2426	2.1931	2.1539	2.122	2.0954	2.0537	2.0095	1.9625	1.9377	1.9119	1.8852	1.8572	1.828	1.7973
15	3.0732	2.6952	2.4898	2.3614	2.273	2.2081	2.1582	2.1185	2.0862	2.0593	2.0171	1.9722	1.9243	1.899	1.8728	1.8454	1.8168	1.7867	1.7551
16	3.0481	2.6682	2.4618	2.3327	2.2438	2.1783	2.128	2.088	2.0553	2.0282	1.9854	1.9399	1.8913	1.8656	1.8388	1.8108	1.7816	1.7508	1.7182
17	3.0262	2.6446	2.4374	2.3078	2.2183	2.1524	2.1017	2.0613	2.0284	2.0009	1.9577	1.9117	1.8624	1.8362	1.809	1.7805	1.7506	1.7191	1.6856
18	3.007	2.624	2.416	2.2858	2.1958	2.1296	2.0785	2.0379	2.0047	1.977	1.9333	1.8868	1.8369	1.8104	1.7827	1.7537	1.7232	1.691	1.6567
19	2.9899	2.6056	2.397	2.2663	2.176	2.1094	2.058	2.0171	1.9836	1.9557	1.9117	1.8647	1.8142	1.7873	1.7592	1.7298	1.6988	1.6659	1.6308
20	2.9747	2.5893	2.3801	2.2489	2.1582	2.0913	2.0397	1.9985	1.9649	1.9367	1.8924	1.8449	1.7938	1.7667	1.7382	1.7083	1.6768	1.6433	1.6074
21	2.961	2.5746	2.3649	2.2333	2.1423	2.0751	2.0233	1.9819	1.948	1.9197	1.875	1.8272	1.7756	1.7481	1.7193	1.689	1.6569	1.6228	1.5862
22	2.9486	2.5613	2.3512	2.2193	2.1279	2.0605	2.0084	1.9668	1.9327	1.9043	1.8593	1.8111	1.759	1.7312	1.7021	1.6714	1.6389	1.6042	1.5668
23	2.9374	2.5493	2.3387	2.2065	2.1149	2.0472	1.9949	1.9531	1.9189	1.8903	1.845	1.7964	1.7439	1.7159	1.6864	1.6554	1.6224	1.5871	1.549
24	2.9271	2.5383	2.3274	2.1949	2.103	2.0351	1.9826	1.9407	1.9063	1.8775	1.8319	1.7831	1.7302	1.7019	1.6721	1.6407	1.6073	1.5715	1.5327
25	2.9177	2.5283	2.317	2.1842	2.0922	2.0241	1.9714	1.9293	1.8947	1.8658	1.82	1.7708	1.7175	1.689	1.659	1.6272	1.5934	1.557	1.5176
26	2.9091	2.5191	2.3075	2.1745	2.0822	2.0139	1.961	1.9188	1.8841	1.855	1.809	1.7596	1.7059	1.6771	1.6468	1.6147	1.5805	1.5437	1.5036
27	2.9012	2.5106	2.2987	2.1655	2.073	2.0045	1.9515	1.9091	1.8743	1.8451	1.7989	1.7492	1.6951	1.6662	1.6356	1.6032	1.5686	1.5313	1.4906
28	2.8939	2.5028	2.2906	2.1571	2.0645	1.9959	1.9427	1.9001	1.8652	1.8359	1.7895	1.7395	1.6852	1.656	1.6252	1.5925	1.5575	1.5198	1.4784
29	2.887	2.4955	2.2831	2.1494	2.0566	1.9878	1.9345	1.8918	1.8568	1.8274	1.7808	1.7306	1.6759	1.6466	1.6155	1.5825	1.5472	1.509	1.467
30	2.8807	2.4887	2.2761	2.1422	2.0493	1.9803	1.9269	1.8841	1.849	1.8195	1.7727	1.7223	1.6673	1.6377	1.6065	1.5732	1.5376	1.4989	1.4564
40	2.8354	2.4404	2.2261	2.091	1.9968	1.9269	1.8725	1.8289	1.7929	1.7627	1.7146	1.6624	1.6052	1.5741	1.5411	1.5056	1.4672	1.4248	1.3769
60	2.7911	2.3933	2.1774	2.041	1.9457	1.8747	1.8194	1.7748	1.738	1.707	1.6574	1.6034	1.5435	1.5107	1.4755	1.4373	1.3952	1.3476	1.2915
120	2.7478	2.3473	2.13	1.9923	1.8959	1.8238	1.7675	1.722	1.6843	1.6524	1.6012	1.545	1.4821	1.4472	1.4094	1.3676	1.3203	1.2646	1.1926
Inf	2.7055	2.3026	2.0838	1.9449	1.8473	1.7741	1.7167	1.6702	1.6315	1.5987	1.5458	1.4871	1.4206	1.3832	1.3419	1.2951	1.24	1.1686	1

شکل 1: جدول توزیع F