

به نام خدا

مهدی فقهی

۴۰۱۷۲۲۱۳۶

پروژه دوم درس یادگیری ماشین

(سوال اول)

به کمک تابع `generate dataset` به شکل های مختلفی دیتا ست خود را می سازیم .

```
def generate_dataset(size, mode):
    if mode == 0: #Simple Ruling data generation
        x = []
        y = []
        target = []
        for i in range(size):
            # class zero
            x.append(np.round(random.uniform(0, 2.5), 1))
            y.append(np.round(random.uniform(0, 20), 1))
            target.append(0)
            # class one
            x.append(np.round(random.uniform(1, 5), 2))
            y.append(np.round(random.uniform(15, 25), 2))
            target.append(1)

    elif mode == 1: #Gaussian Quantiles
        X1, t1 = make_gaussian_quantiles(cov=3., n_samples=size, n_features=2, n_classes=2, random_state=1)

        X1 = pd.DataFrame(X1, columns=['x', 'y'])
        x = X1["x"]
        x=x.to_numpy()
        y = X1["y"]
        y=y.to_numpy()
        t1 = pd.Series(t1)
        target=t1.to_numpy()

    elif mode == 2: #Blobs
        X, t1 = make_blobs(n_samples=size, centers=2, n_features=2, random_state=17)

        X = pd.DataFrame(X, columns=['x', 'y'])
        x = X["x"]
        x=x.to_numpy()
        y = X["y"]
        y=y.to_numpy()
```

سپس یک تابع دیگر می‌نویسیم به اسم experiment model که براساس دیتاست ساخته شده انواع مختلف هسته و پارامترهای مربوط به آن را می‌آزماید و بهترین پارامتر را برای هر کدام از هسته‌ها پیدا می‌کند .

```
def experiment_model(X_train,y_train,y,f1,f2,y_test,X_test):
    result_list = []
    # Define the parameter grid for gamma and C
    param_grid = {'gamma': [0.1, 1, 10], 'C': [0.1, 1, 10]}

    # Initialize the SVM with a rbf kernel
    svm_model = SVC(kernel='rbf')
    clf_one = GridSearchCV(estimator=svm_model, param_grid=param_grid)
    clf_one.fit(X_train, y_train)
    Predicted_labels= clf_one.predict(X_test)
    result_list.append((Predicted_labels,clf_one))
    print(f"SVM , Kernel = RBF , Best parameters: {clf_one.best_params_}")

    # Define the parameter grid for gamma and C
    param_grid = {'C': [0.1, 1, 10]}
    svm_model = SVC(kernel='linear')
    clf_two = GridSearchCV(estimator=svm_model, param_grid=param_grid)
    clf_two.fit(X_train, y_train)
    Predicted_labels= clf_two.predict(X_test)
    result_list.append((Predicted_labels,clf_two))
    print(f"SVM , Kernel = Linear ,Best parameters: {clf_two.best_params_}")

    # Initialize the SVM with a polynomial kernel
    param_grid = {"degree": [2, 3, 4],"C": [ 0.1, 1, 10]}
    svm_model = SVC(kernel='poly')
    clf_five = GridSearchCV(estimator=svm_model, param_grid=param_grid)
    clf_five.fit(X_train, y_train)
    Predicted_labels= clf_five.predict(X_test)
    result_list.append((Predicted_labels,clf_five))
    print(f"SVM , Kernel = Poly , Best parameters: {clf_five.best_params_}")

    param_grid = {'gamma': [0.1, 1, 10], 'C': [0.1, 1, 10]}
    # Initialize the SVM with a Sigmoid kernel
    svm_model = SVC(kernel='sigmoid')
    clf_six = GridSearchCV(estimator=svm_model, param_grid=param_grid)
    clf_six.fit(X_train, y_train)
    Predicted_labels= clf_six.predict(X_test)
    result_list.append((Predicted_labels,clf_six))
    print(f"SVM , Kernel = Sigmoid , Best parameters: {clf_six.best_params_}")

    print("SVM , Kernel = Manual, ")

    # Create an SVM model with Polynomial kernel
    clf_seven = SVC(kernel=polynomial_kernel)

    # Fit the model to your data
    clf_seven.fit(X_train, y_train)
    Predicted_labels= clf_seven.predict(X_test)
    result_list.append((Predicted_labels,clf_seven))

    return result_list
```

علاوه بر کرنل‌های مرسوم یک کرنل manual نیز اضافه کرده‌ام که همان کرنلی هست که باید خودم می‌نوشتم .

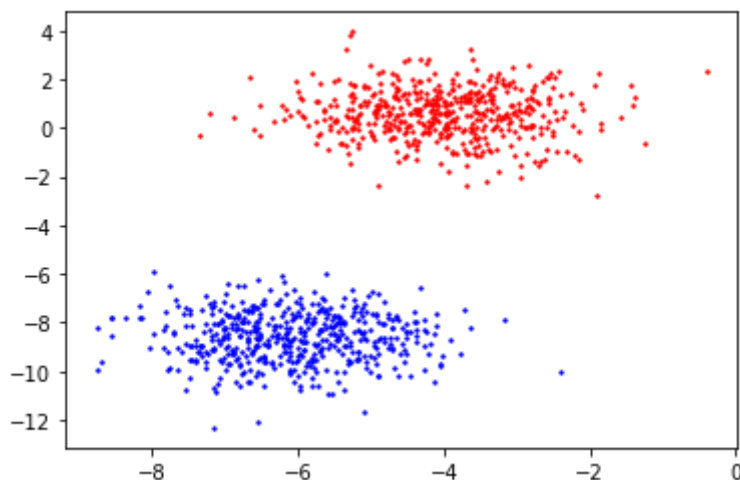
```
def polynomial_kernel(X, Y, degree=3, gamma=1, coef0=1):
    """
    Polynomial kernel implementation for SVM using NumPy.

    Args:
        X (ndarray): Input data with shape (n_samples_X, n_features).
        Y (ndarray): Input data with shape (n_samples_Y, n_features).
        degree (int): Degree of the polynomial kernel function.
        gamma (float): Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
        coef0 (float): Independent term in kernel function. It is only significant
            in 'poly' and 'sigmoid'.

    Returns:
        K (ndarray): Computed kernel matrix with shape (n_samples_X, n_samples_Y).
    """
    K = np.dot(X, Y.T)
    K = (gamma * K + coef0) ** degree

    return K
```

در ادامه شروع کردم به بررسی نتایج بر روی مدل‌های دیتاست که ساختم .
شکل‌ها را براساس ساده به سخت در گزارش قرار می‌دهم .
(شکل اول)



هسته‌ها و بهترین پارامترهایشان:

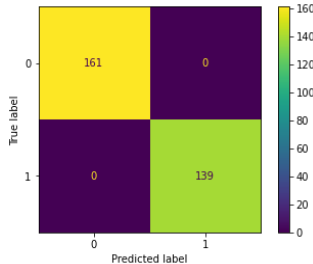
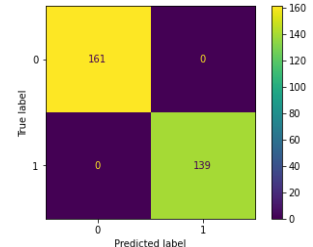
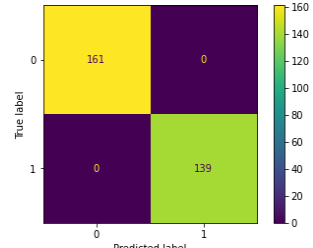
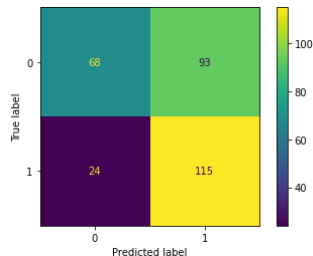
{SVM , Kernel = RBF , Best parameters: {'C': 0.1, 'gamma': 0.1

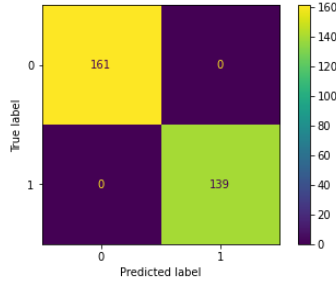
{SVM , Kernel = Linear ,Best parameters: {'C': 0.1

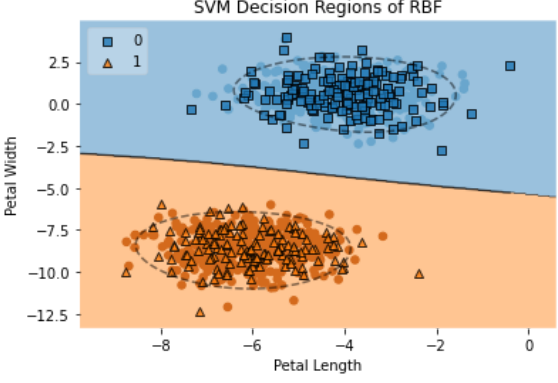
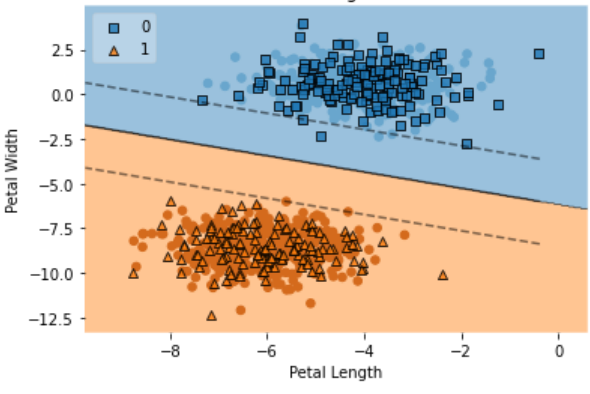
{SVM , Kernel = Poly , Best parameters: {'C': 0.1, 'degree': 2

{SVM , Kernel = Sigmoid , Best parameters: {'C': 0.1, 'gamma': 0.1

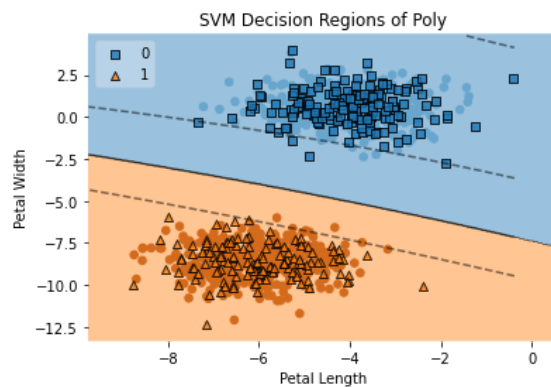
نتائج :

	precision	recall	f1	acc	confusion matrix
rbf	1	1	1	1	
linear	1	1	1	1	
poly					
sigmoid	0.61	0.8	0.5	0.6	
manual	1	1	1	1	

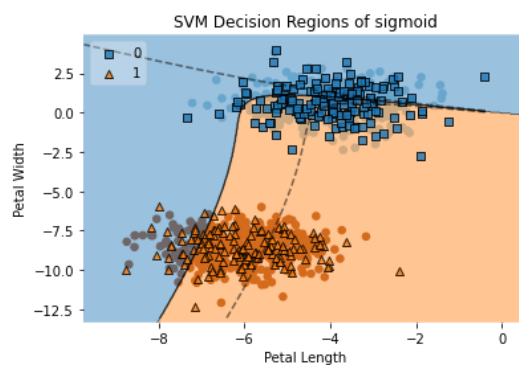
					 <p>Confusion matrix for SVM RBF model:</p> <table><tr><th></th><th>Predicted label 0</th><th>Predicted label 1</th></tr><tr><th>True label 0</th><td>161</td><td>0</td></tr><tr><th>True label 1</th><td>0</td><td>139</td></tr></table>		Predicted label 0	Predicted label 1	True label 0	161	0	True label 1	0	139
	Predicted label 0	Predicted label 1												
True label 0	161	0												
True label 1	0	139												

rbf	 <p>SVM Decision Regions of RBF</p> <p>The plot shows two classes of data points (blue squares for class 0 and orange triangles for class 1) in a 2D space defined by Petal Length (x-axis) and Petal Width (y-axis). The decision boundary is a non-linear curve that separates the two classes. The regions are shaded light blue for class 0 and light orange for class 1.</p>
linear	 <p>SVM Decision Regions of linear</p> <p>The plot shows the same two classes of data points in a 2D space. The decision boundary is a straight line, indicating a linear separation between the two classes. The regions are shaded light blue for class 0 and light orange for class 1.</p>

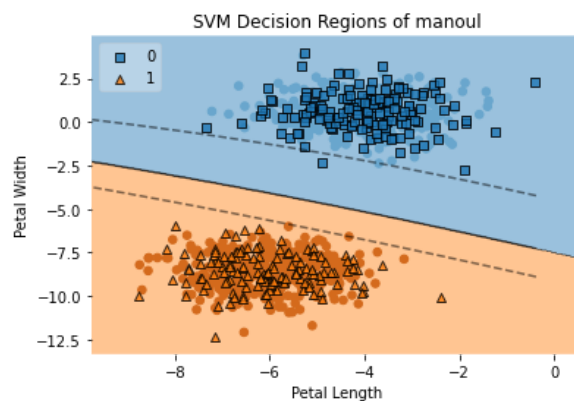
poly



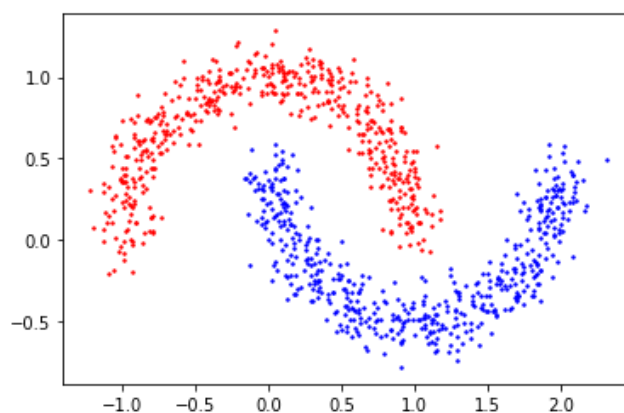
sigmoid



manual



شكل دوم)



{SVM , Kernel = RBF , Best parameters: {'C': 0.1, 'gamma': 10

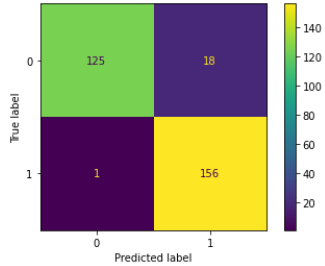
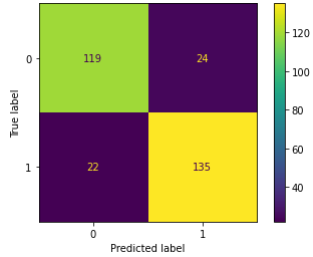
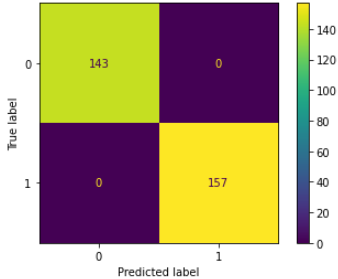
{SVM , Kernel = Linear ,Best parameters: {'C': 1

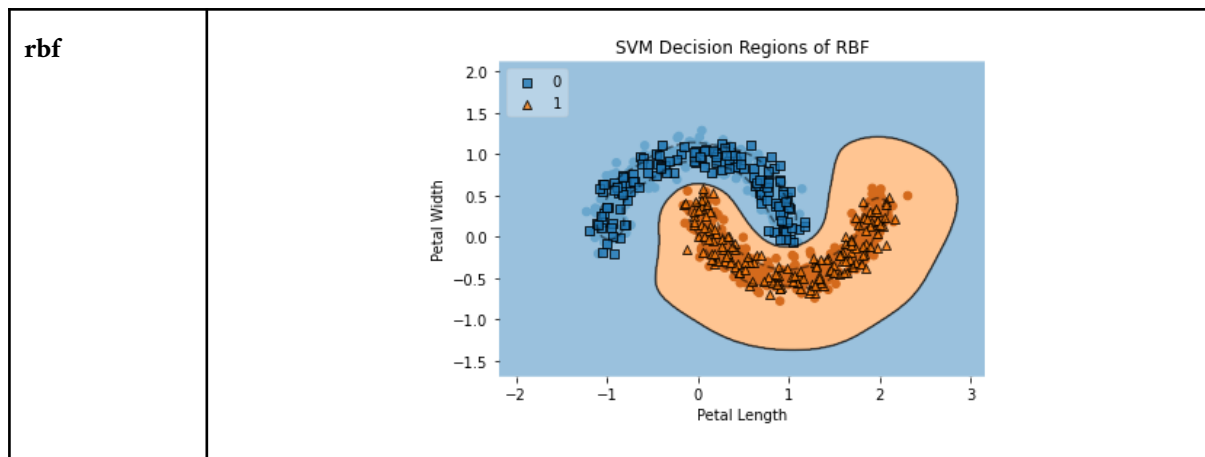
{SVM , Kernel = Poly , Best parameters: {'C': 0.1, 'degree': 3

{SVM , Kernel = Sigmoid , Best parameters: {'C': 1, 'gamma': 0.1

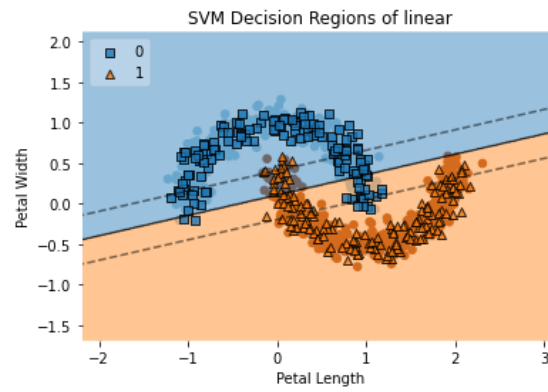
نتائج :

	precision	recall	accuracy	f1	conf matrix
rbf	1	1	1	1	
linear	0.86	0.87	0.86	0.86	

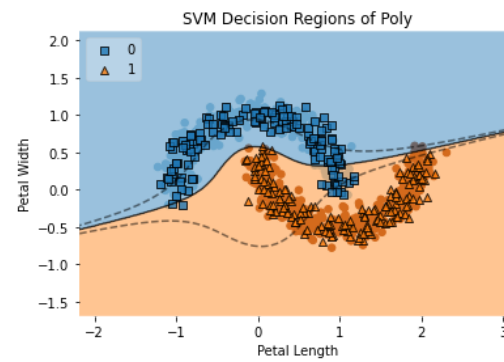
poly	0.89	0.99	0.93	0.94	
sigmoid	0.84	0.85	0.84	0.85	
manual	1	1	1	1	



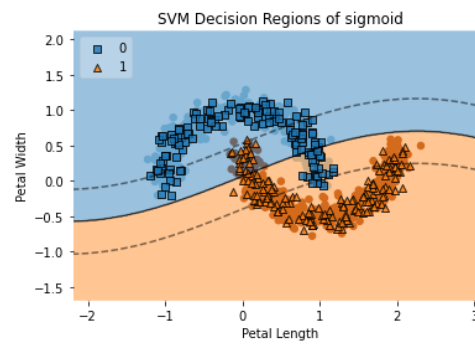
linear



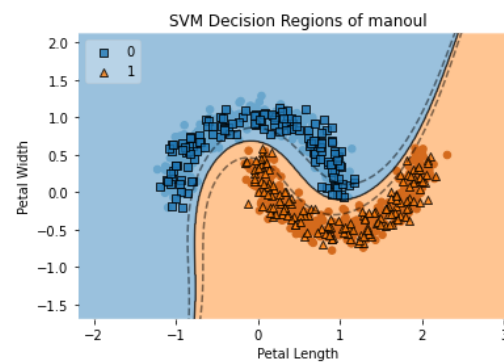
poly



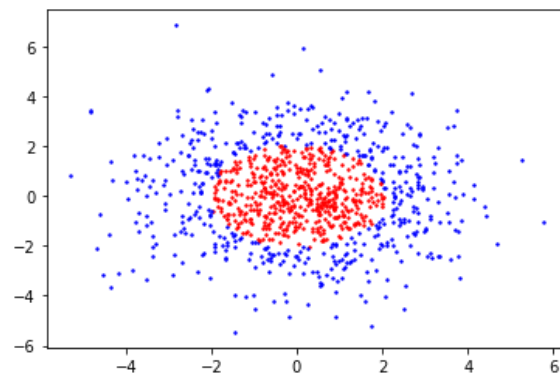
sigmoid



manual



شکل سوم)



هسته‌ها و بهترین پارامترهایشان :

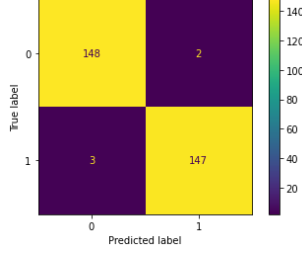
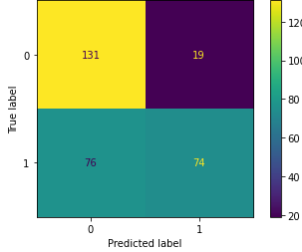
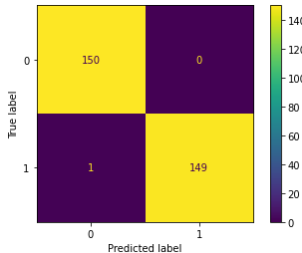
{SVM , Kernel = RBF , Best parameters: {'C': 10, 'gamma': 0.1

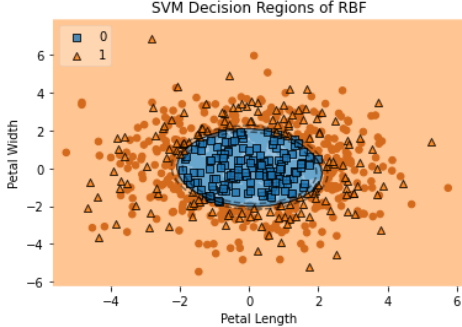
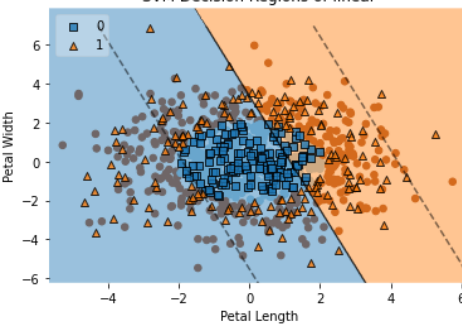
{SVM , Kernel = Linear ,Best parameters: {'C': 0.1

{SVM , Kernel = Poly , Best parameters: {'C': 10, 'degree': 2

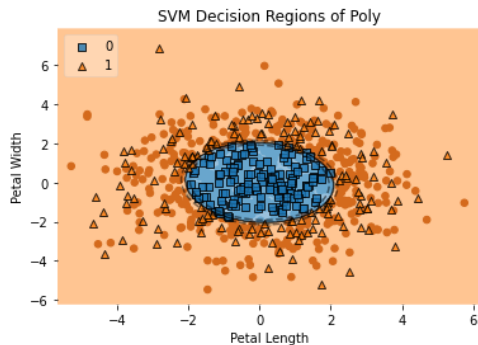
{SVM , Kernel = Sigmoid , Best parameters: {'C': 1, 'gamma': 0.1

	precision	recall	accuracy	f1	conf matrix
rbf	0.98	0.98	0.98	0.98	
linear	0.71	0.38	0.61	0.5	

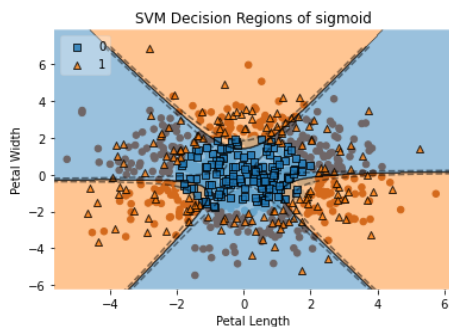
poly	0.98	0.98	0.98	0.98	
sigmoid	0.79	0.49	0.68	0.60	
manual	1	0.99	0.99	0.99	

rbf					
linear					

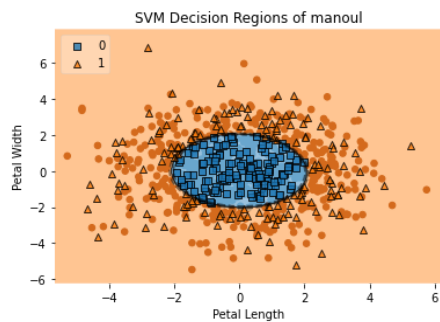
poly



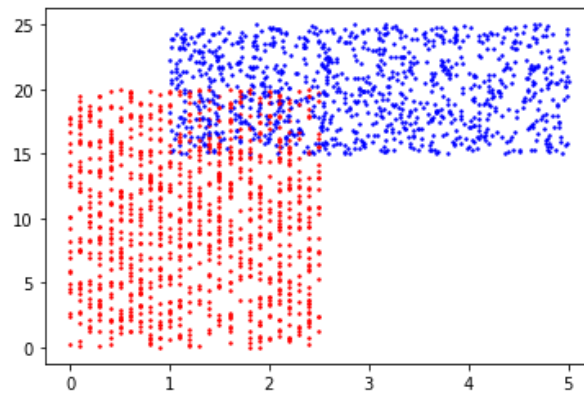
sigmoid



manual



شکل چهارم)



هسته ها و بهترین پارامترهایشان :

{SVM , Kernel = RBF , Best parameters: {'C': 10, 'gamma': 1

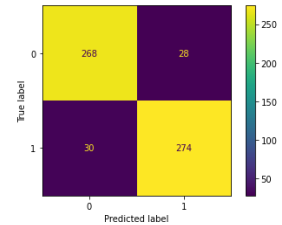
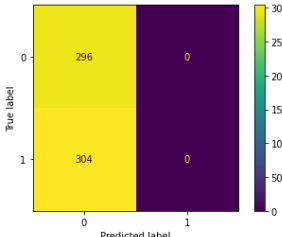
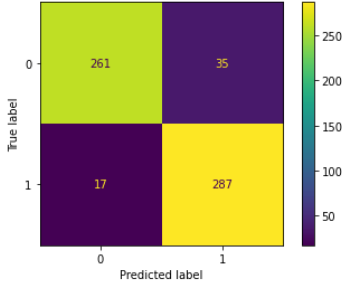
{SVM , Kernel = Linear ,Best parameters: {'C': 10

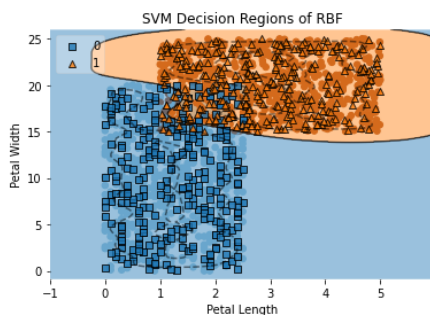
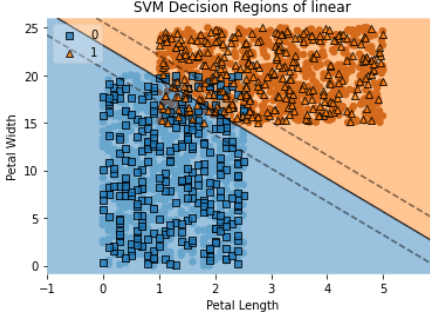
{SVM , Kernel = Poly , Best parameters: {'C': 1, 'degree': 2

{SVM , Kernel = Sigmoid , Best parameters: {'C': 0.1, 'gamma': 10

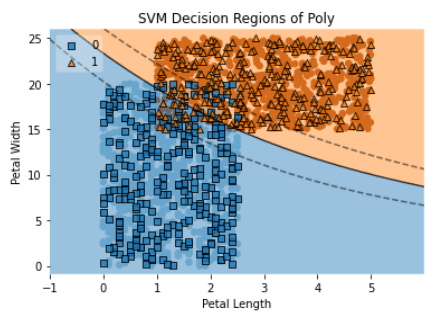
نتایج :

	accuracy	recall	precision	f1	conf matrix
rbf	0,90	0,93	0,90	0,92	
linear	0,90	0,91	0,90	0,90	

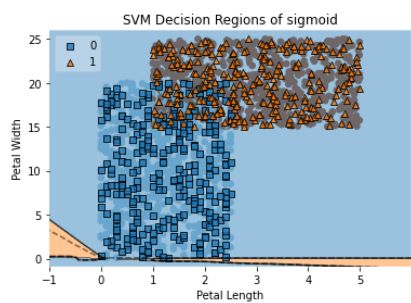
poly	0.90	0.90	0.90	0.90	
sigmoid	0.49	0	0	0	
manual	0.91	0.94	0.89	0.91	

rbf	
linear	

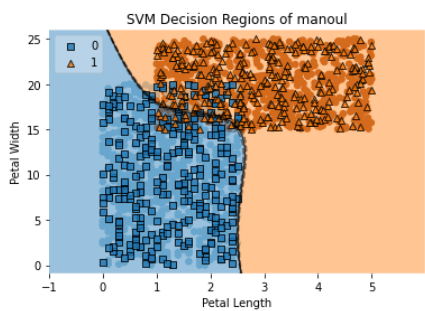
poly



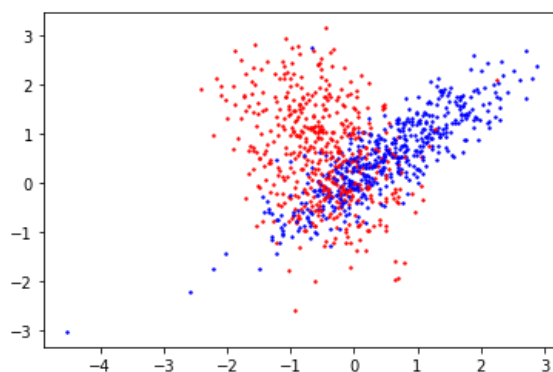
sigmoid



manual



شکل پنچ



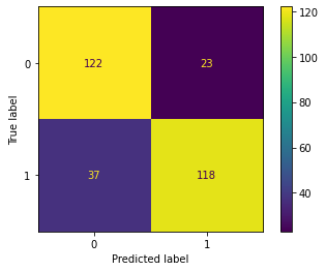
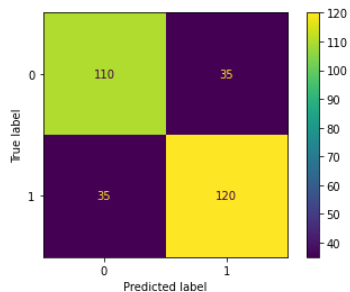
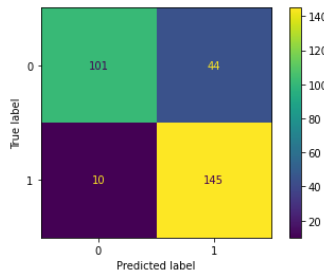
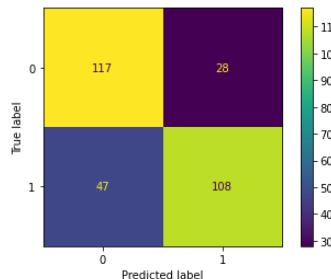
هسته‌ها و بهترین پارامترهایشان :

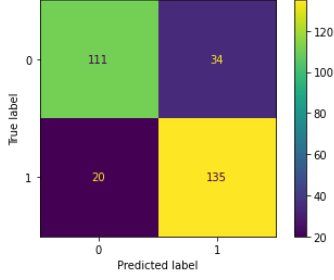
{SVM , Kernel = RBF , Best parameters: {'C': 1, 'gamma': 1

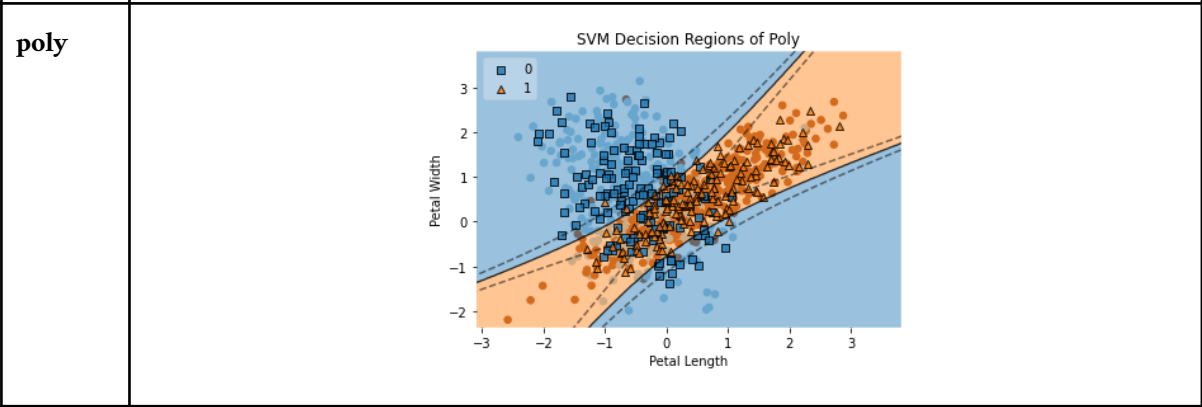
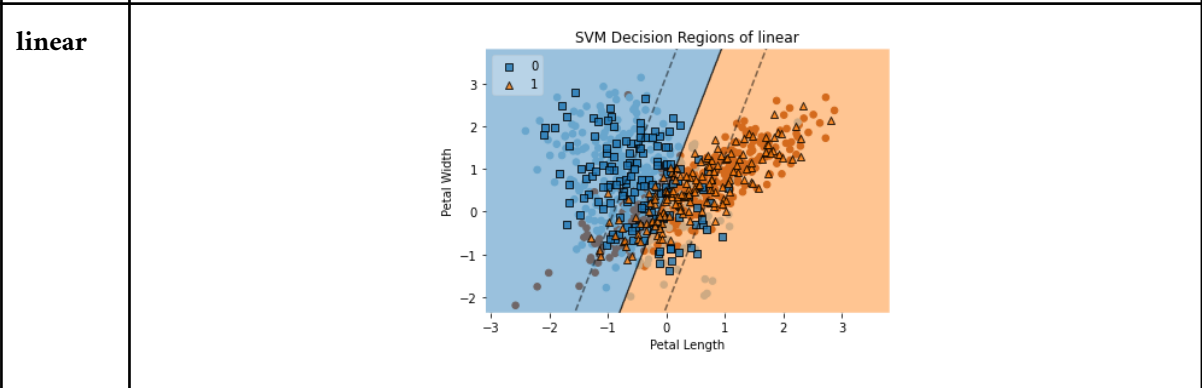
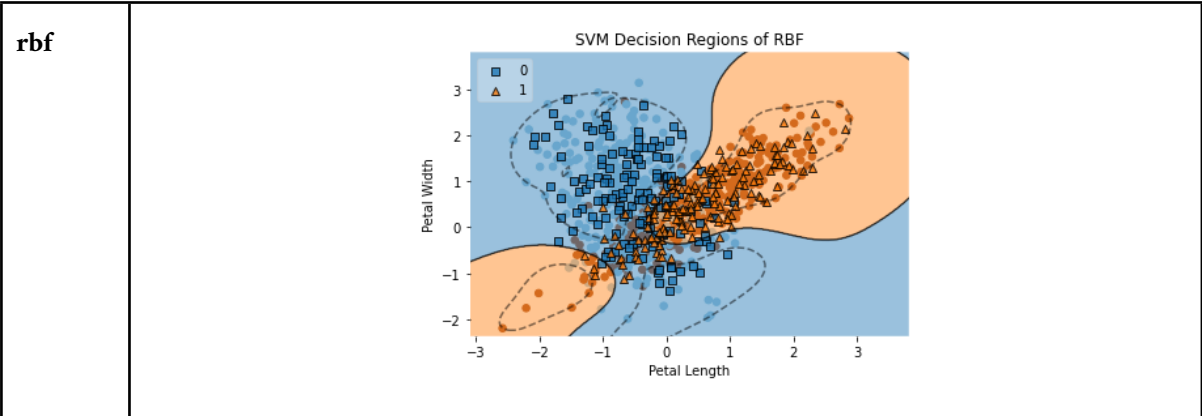
{SVM , Kernel = Linear ,Best parameters: {'C': 0.1

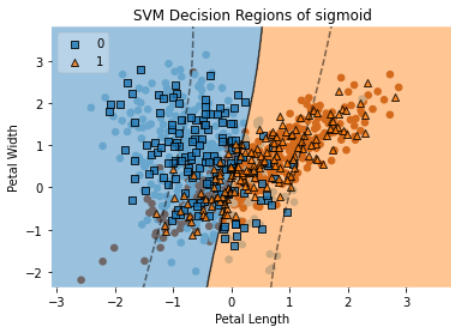
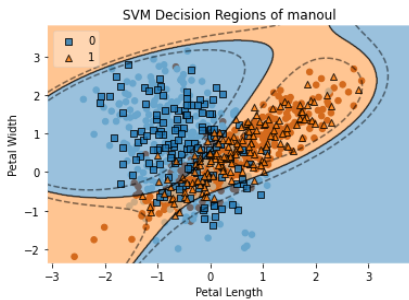
{SVM , Kernel = Poly , Best parameters: {'C': 10, 'degree': 2

{SVM , Kernel = Sigmoid , Best parameters: {'C': 0.1, 'gamma': 0.1

	accuracy	recall	precision	f1	conf matrix
rbf	0.8	0.76	0.83	0.79	
linear	0.76	0.77	0.77	0.77	
poly	0.82	0.93	0.76	0.84	
sigmoid	0.75	0.69	0.79	0.74	

manual	0.82	0.87	0.79	0.83	
---------------	-------------	-------------	-------------	-------------	--



sigmoid	 <p>The plot titled 'SVM Decision Regions of sigmoid' shows two classes of data points (0 and 1) separated by a linear decision boundary. The x-axis is 'Petal Length' and the y-axis is 'Petal Width'. The background is shaded blue for class 0 and orange for class 1.</p>
manual	 <p>The plot titled 'SVM Decision Regions of manoul' shows two classes of data points (0 and 1) separated by a non-linear decision boundary. The x-axis is 'Petal Length' and the y-axis is 'Petal Width'. The background is shaded blue for class 0 and orange for class 1.</p>

قسمت پاسخ به پرسش‌ها قسمت اول:

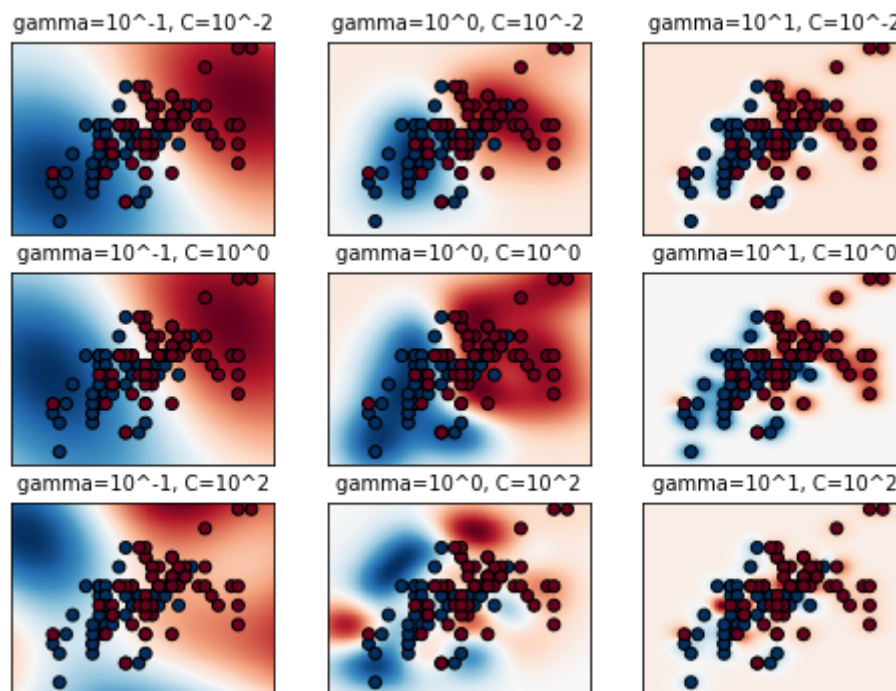
-رفته رفته پیچیدگی داده‌ها را بیشتر کرده و عملیات بالا را روی داده‌ها تکرار کنید. پیچیدگی داده‌ها چه تاثیری در انتخاب هسته و پارامترهای مربوطه خواهند داشت؟ به طور کامل شرح دهید.

خوب طبیعی است که هرچه داده‌های ما پیچیده‌تر شوند نیاز به هسته پیچیده‌تری برای حل مسئله مورد نظر است . این مسئله به خوبی با این پنج شکلی که از آسون به سخت کشیده شده مشخص است و همانطور که مشاهده می‌کنید دقت تابع خطی هرچه می‌گذرد در مراحل بعد کمتر و کمتر می‌شود و به ازای آن خوبی کارکرد بایک کرنل پیچیده‌تر نیز دیده می‌شود هرچند همانطور که دیده شد بعضی از کرنل‌های به ظاهر سخت مانند sigmoid اگر در جای خود همانند شکل چهار و پنج درست استفاده نشوند و برای دادگان ساده از آنان استفاده شود ممکن است چه بسا دارای خطای بیشتری از کرنل‌های ساده‌تر باشند و آن برازش عملاً برازش مناسبی برای دادگان نباشد . پارامترها نیز همانطور که مشاهده می‌فرمایید با افزایش سختی مدل C افزایش پیدا می‌کند و همچنین gamma نیز افزایش پیدا می‌کند همین اتفاق برای degree در مدل poly نیز اتفاق می‌افتد .

-آیا می توان به طور قطع در مورد نوع هسته خاصی اظهار نظر کرد و مطمئن بود همواره بهترین خواهد بود؟ به طور کامل شرح دهید.

خیر. بهترین هسته و پارامترهای آن باید با آزمایش بدست بیاید و این مهم به این علت است که دادگان مختلف رفتارهای مختلفی از خود نشان می دهند که برای پیدا کردن مرز تفکیک بین آنها نیاز است که هسته های مختلف را نسبت به آنان بسنجیم و با بررسی معیارهای آموزش بسنجیم کدام مدل بهترین نماینده برای این نوع دادگان است همچنین گاهی پیش میاید که دو مدل یکی با سختی بالا و یک مدل ساده تر یک نتیجه را می گیرند مانند شکل شماره یک اینجا اولویت با انتخاب مدل ساده تر است پس نمی توانیم به طور یقین برای تمامی مسائل مثلاً از rbf استفاده کرد برای مثال شکل آخر برای رد این ادعا کافی است زیرا مدل sigmoid , poly به مراتب نتایج بهتری از rbf گرفتند.

- در مورد پارامتر تنظیم روی داده ها توضیح دهید و نتایج مختلف را به سبب تغییر میزان آن به طور منحصر به فرد به غیر از مابقی پارامترها به طور کامل شرح دهید.



پارامتر تنظیم یا regularization term در فرمول اصلی که به شکل زیر تعریف می شود برابر با ۸ است.

Computing the (soft-margin) SVM classifier amounts to min

$$\left[\frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i - b)) \right] + \lambda \|\mathbf{w}\|^2.$$

حال آنکه در فرمول sklearn برابر است با :

$$C \sum_{i=1,n} \mathcal{L}(f(x_i), y_i) + \Omega(w)$$

ترم C عملاً معکوس γ است . طبق فرمول بالا هرچه C بزرگتر شود میزان loss یا خطا تاثیرش در مدل اصلی بیشتر می شود یعنی تحمل مدل نسبت به خطا خیلی کاهش پیدا می کند و سعی می کند تعداد خطا را کمینه کند . همانطور که در شکل بالا می بینید تاثیر C به خوبی در کنار gamma مشخص است . با فرض ثابت در نظر گرفتن gamma در یک مدل rbf و دقت کردن به اینکه افزایش C چه تاثیری در مدل می گذارد ، خواهیم دید که باعث کاهش generalization به جهت کاهش میزان ارور می شود کاهش فضای آبی رنگ و قرمز رنگ به معنای کاهش یافتن قدرت تعمیم در برابر کاهش میزان خطا است .

سوال دوم)

برای سوال دوم از مجموعه دادگان Speed Dating Experiment استفاده کردم . این مجموعه دادگان که شامل ۱۹۵ ستون و ۸۳۷۸ داده است .

داده ها از شرکت کنندگان در رویدادهای دوستیابی experimental speed dating از 2002-2004 جمع آوری شد. در طول رویدادها، شرکت کنندگان چهار دقیقه با هر شرکت کننده دیگری از جنس مخالف صحبت داشتند. در پایان چهار دقیقه از شرکت کنندگان پرسیده شد که آیا مایلند دوباره شخص مقابل خود را ببینند یا نه . همچنین از آنها خواسته شد که شخص مورد نظر را بر اساس شش ویژگی رتبه بندی کنند: جذابیت، صداقت، هوش، سرگرمی، جاه طلبی و علائق مشترک. مجموعه داده همچنین شامل داده های پرسشنامه است که از شرکت کنندگان در نقاط مختلف فرآیند جمع آوری شده است. این زمینه ها عبارتند از: جمعیت شناسی، عادات دوستیابی، درک خود از ویژگی های کلیدی، باورها در مورد آنچه دیگران در یک همسر ارزشمند می دانند و اطلاعات مربوط به شیوه زندگی.

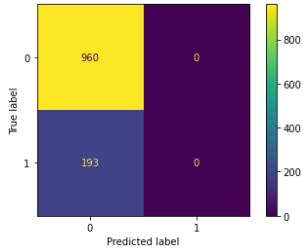
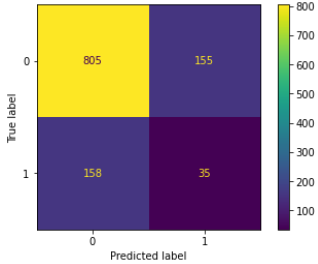
این دادگان دارای مقادیر null است همچنین یک سری ستون دارد که با ستون match که ستونی هست که می خواهیم classification را انجام دهیم ارتباط و همبستگی معنایی بالایی دارند که این ستون ها را نیز برای یادگیری واقعی تر باید حذف کنیم . موارد بالا و یک سری موارد دیگر شامل یافتن featureهای جدید براساس داده های را در عملیات preprocess که در کد آمده است انجام می دهیم . ((موارد انجام شده از preprocess طبق یک کد آماده در github انجام شده است .))
مهمترین چالش این دادگان این است که ۸۳ درصد موارد برچسب match برابر صفر است و در 16 درصد موارد برابر با یک است که عملاً متناسب با خواست سوال یک مجموعه داده unbalanced در اختیار داریم .
پس از انجام عملیات preprocess می رویم سراغ عملیات اصلی مطلوب classification .
همانند سوال اول از تابع experiment model برای پیدا کردن بهترین مدل استفاده می کنیم .
در این اطلاعات به دلیل گستردگی و ابعاد بالایی که داشت هسته های linear , poly توانایی converge کردن نداشتن و فقط ما می توانستیم از هسته های sigmoid , rbf برای ارزیابی استفاده کنیم .

مشاهده نتایج :

ساده ترین شکل ممکن برای پیاده سازی با SVM (فقط مدل با هسته های زیر توانایی converge کردن داشتند)

SVM , Kernel = RBF , Best parameters: 'C': 0.01, 'gamma': 0.0

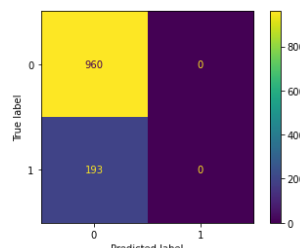
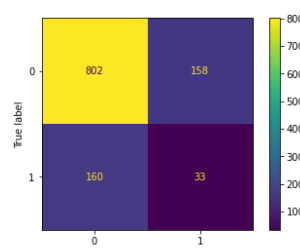
SVM , Kernel = Sigmoid , Best parameters: 'C': 1, 'gamma': 0.01

	accuracy	recall	precision	f1	conf matrix
rbf	0.83	0	0	0	
sigmoid	0.72	0.18	0.18	0.18	

پیاده سازی با PCA 100 :

SVM , Kernel = RBF , Best parameters: 'C': 0.01, 'gamma'

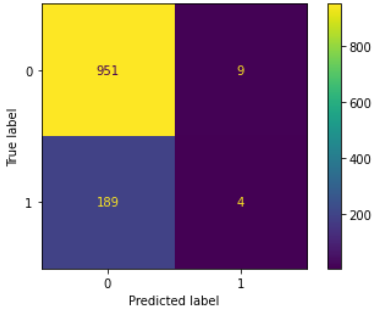
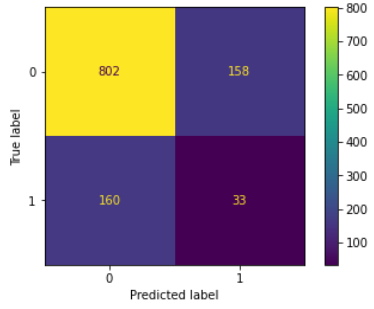
SVM , Kernel = Sigmoid , Best parameters: 'C': 10, 'gamma': 0.01

	accuracy	recall	precision	f1	conf matrix									
rbf	0.83	0	0	0	 <table><tr><th>True label \ Predicted label</th><th>0</th><th>1</th></tr><tr><th>0</th><td>960</td><td>0</td></tr><tr><th>1</th><td>193</td><td>0</td></tr></table>	True label \ Predicted label	0	1	0	960	0	1	193	0
True label \ Predicted label	0	1												
0	960	0												
1	193	0												
sigmoid	0.72	0.17	0.17	0.17	 <table><tr><th>True label \ Predicted label</th><th>0</th><th>1</th></tr><tr><th>0</th><td>802</td><td>158</td></tr><tr><th>1</th><td>160</td><td>33</td></tr></table>	True label \ Predicted label	0	1	0	802	158	1	160	33
True label \ Predicted label	0	1												
0	802	158												
1	160	33												

پیاده سازی PCA 30 :

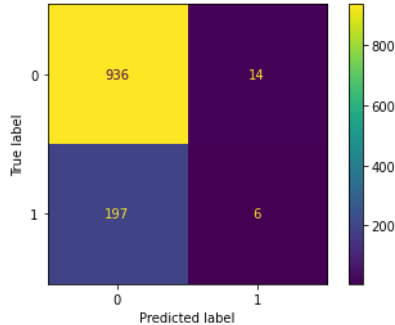
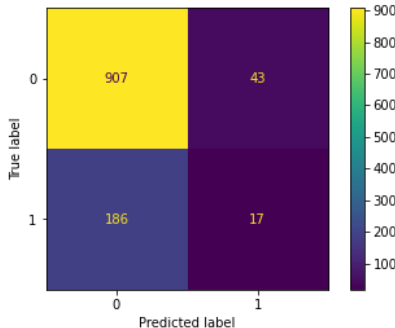
SVM , Kernel = RBF , Best parameters: 'C': 10, 'gamma': 0.01

SVM , Kernel = Sigmoid , Best parameters: 'C': 10, 'gamma': 0.01

	accuracy	recall	precision	f1	conf matrix
rbf	0.82	0.02	0.3	0.04	 <p>True label</p> <p>Predicted label</p>
sigmoid	0.72	0.17	0.17	0.17	 <p>True label</p> <p>Predicted label</p>

SVM , Kernel = RBF , Best parameters: 'C': 100, 'gamma': 10

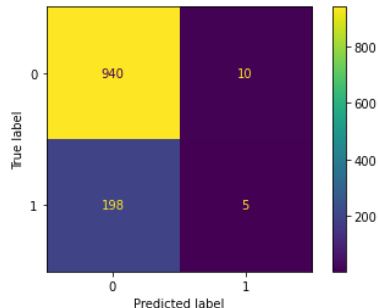
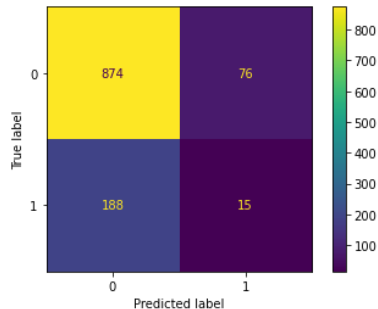
SVM , Kernel = Sigmoid , Best parameters: 'C': 100, 'gamma': 10

	accuracy	recall	precision	f1	confusion matrix									
rbf	0.82	0.024	0.33	0.05	 <table><tr><th>True label \ Predicted label</th><th>0</th><th>1</th></tr><tr><th>0</th><td>936</td><td>14</td></tr><tr><th>1</th><td>197</td><td>6</td></tr></table>	True label \ Predicted label	0	1	0	936	14	1	197	6
True label \ Predicted label	0	1												
0	936	14												
1	197	6												
sigmoid	0.8	0.08	0.28	0.12	 <table><tr><th>True label \ Predicted label</th><th>0</th><th>1</th></tr><tr><th>0</th><td>907</td><td>43</td></tr><tr><th>1</th><td>186</td><td>17</td></tr></table>	True label \ Predicted label	0	1	0	907	43	1	186	17
True label \ Predicted label	0	1												
0	907	43												
1	186	17												

پیاده سازی LLE 200 :

SVM , Kernel = RBF , Best parameters: 'C': 100, 'gamma': 10

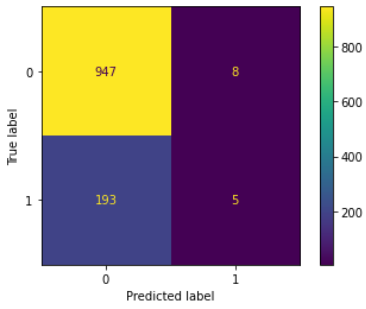
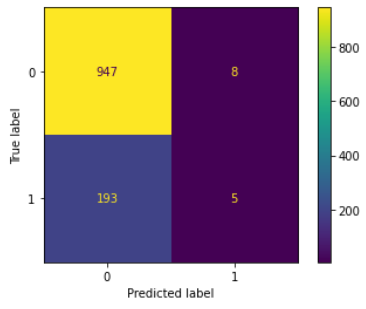
SVM , Kernel = Sigmoid , Best parameters: 'C': 10, 'gamma': 10

	accuracy	recall	precision	f1	conf matrix
rbf	0.82	0.025	0.33	0.045	
sigmoid	0.77	0.07	0.16	0.10	

پیاده سازی LLE 30 :

SVM , Kernel = RBF , Best parameters: 'C': 100, 'gamma': 10

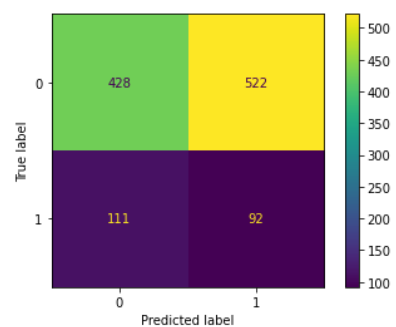
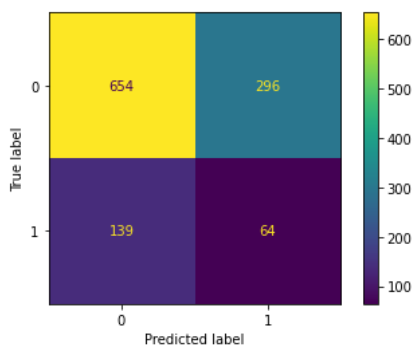
SVM , Kernel = Sigmoid , Best parameters: 'C': 100, 'gamma': 10

	accuracy	recall	precision	f1	conf matrix									
rbf	0.82	0.025	0.38	0.05	 <table><tr><th>True \ Pred</th><th>0</th><th>1</th></tr><tr><th>0</th><td>947</td><td>8</td></tr><tr><th>1</th><td>193</td><td>5</td></tr></table>	True \ Pred	0	1	0	947	8	1	193	5
True \ Pred	0	1												
0	947	8												
1	193	5												
sigmoid	0.77	0.06	0.1	0.08	 <table><tr><th>True \ Pred</th><th>0</th><th>1</th></tr><tr><th>0</th><td>947</td><td>8</td></tr><tr><th>1</th><td>193</td><td>5</td></tr></table>	True \ Pred	0	1	0	947	8	1	193	5
True \ Pred	0	1												
0	947	8												
1	193	5												

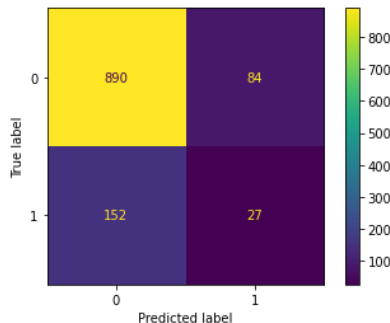
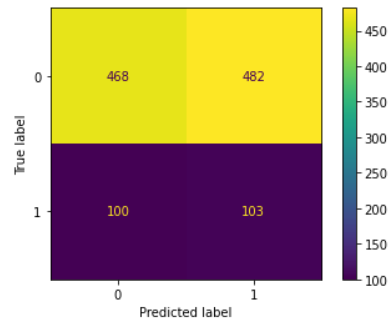
پایاده‌سازی مدل با تغییر وزن به نفع مدل با پراکندگی کمتر :

Best parameters: 'C': 1, 'gamma': 0.001, 'kernel': 'sigmoid

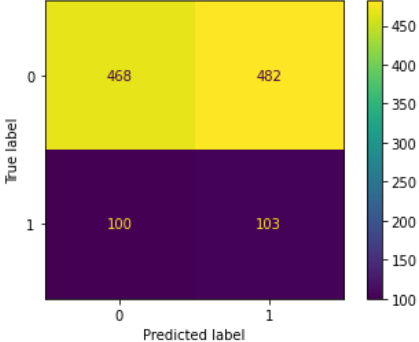
Best parameters: 'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf

	accuracy	recall	precision	f1	conf matrix									
rbf	0.5	0.5	0.17	0.26	 <table><tr><th></th><th>True label 0</th><th>True label 1</th></tr><tr><th>Predicted label 0</th><td>428</td><td>111</td></tr><tr><th>Predicted label 1</th><td>522</td><td>92</td></tr></table>		True label 0	True label 1	Predicted label 0	428	111	Predicted label 1	522	92
	True label 0	True label 1												
Predicted label 0	428	111												
Predicted label 1	522	92												
sigmoid	0.62	0.3	0.17	0.2	 <table><tr><th></th><th>True label 0</th><th>True label 1</th></tr><tr><th>Predicted label 0</th><td>654</td><td>139</td></tr><tr><th>Predicted label 1</th><td>296</td><td>64</td></tr></table>		True label 0	True label 1	Predicted label 0	654	139	Predicted label 1	296	64
	True label 0	True label 1												
Predicted label 0	654	139												
Predicted label 1	296	64												

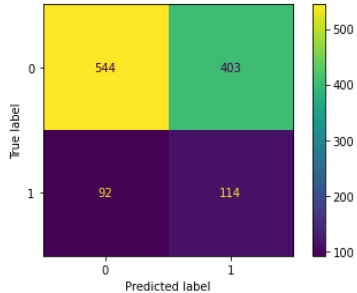
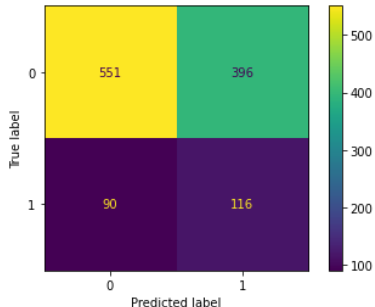
پیاده‌سازی با تغییر وزن به نفع برجسب با تعداد کم PCA 100 :

	accuracy	precision	recall	f1	confusion matrix									
rbf	0.8	0.15	0.25	0.2	 <table><tr><th></th><th>True label 0</th><th>True label 1</th></tr><tr><th>Predicted label 0</th><td>890</td><td>152</td></tr><tr><th>Predicted label 1</th><td>84</td><td>27</td></tr></table>		True label 0	True label 1	Predicted label 0	890	152	Predicted label 1	84	27
	True label 0	True label 1												
Predicted label 0	890	152												
Predicted label 1	84	27												
sigmoid	0.5	0.17	0.5	0.26	 <table><tr><th></th><th>True label 0</th><th>True label 1</th></tr><tr><th>Predicted label 0</th><td>468</td><td>100</td></tr><tr><th>Predicted label 1</th><td>482</td><td>103</td></tr></table>		True label 0	True label 1	Predicted label 0	468	100	Predicted label 1	482	103
	True label 0	True label 1												
Predicted label 0	468	100												
Predicted label 1	482	103												

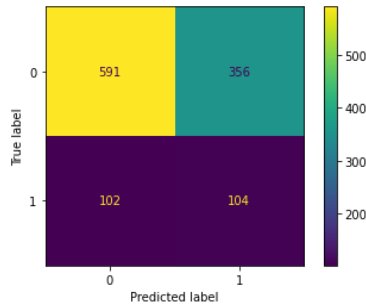
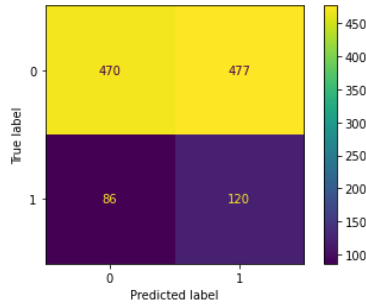
پیاده‌سازی با تغییر وزن به نفع برجسب با تعداد کم PCA 30 :

	accuracy	precision	recall	f1	confusion matrix									
sigmoid	0.5	0.17	0.5	0.30	 <table><thead><tr><th></th><th>0</th><th>1</th></tr></thead><tbody><tr><th>0</th><td>468</td><td>482</td></tr><tr><th>1</th><td>100</td><td>103</td></tr></tbody></table>		0	1	0	468	482	1	100	103
	0	1												
0	468	482												
1	100	103												

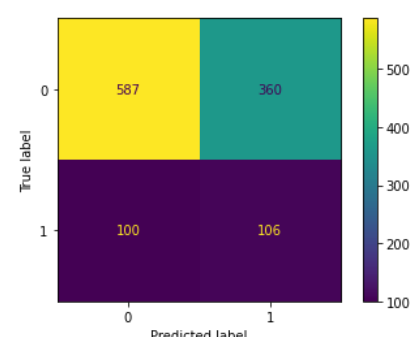
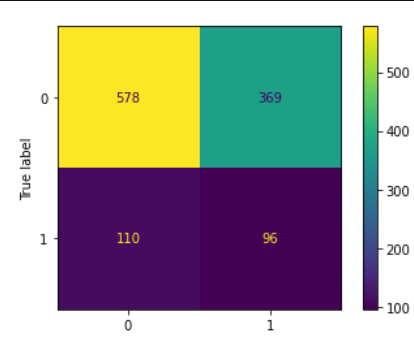
پیاده‌سازی با تغییر وزن به نفع برچسب با تعداد کم 300 LLE:

	accuracy	recall	precision	f1	confusion matrix									
rbf	0.57	0.55	0.22	0.3	 <table><tr><th></th><th>0</th><th>1</th></tr><tr><th>0</th><td>544</td><td>403</td></tr><tr><th>1</th><td>92</td><td>114</td></tr></table>		0	1	0	544	403	1	92	114
	0	1												
0	544	403												
1	92	114												
sigmoid	0.57	0.56	0.22	0.3	 <table><tr><th></th><th>0</th><th>1</th></tr><tr><th>0</th><td>551</td><td>396</td></tr><tr><th>1</th><td>90</td><td>116</td></tr></table>		0	1	0	551	396	1	90	116
	0	1												
0	551	396												
1	90	116												

پیاده سازی با تغییر وزن به نفع برچسب با تعداد کم 200 LLE:

	accuracy	recall	precision	f1	conf matrix									
rbf	0.6	0.5	0.22	0.31	 <p>Confusion matrix for rbf model:</p> <table><tr><th></th><th>Predicted label 0</th><th>Predicted label 1</th></tr><tr><th>True label 0</th><td>591</td><td>356</td></tr><tr><th>True label 1</th><td>102</td><td>104</td></tr></table>		Predicted label 0	Predicted label 1	True label 0	591	356	True label 1	102	104
	Predicted label 0	Predicted label 1												
True label 0	591	356												
True label 1	102	104												
sigmoid	0.51	0.58	0.20	0.3	 <p>Confusion matrix for sigmoid model:</p> <table><tr><th></th><th>Predicted label 0</th><th>Predicted label 1</th></tr><tr><th>True label 0</th><td>470</td><td>477</td></tr><tr><th>True label 1</th><td>86</td><td>120</td></tr></table>		Predicted label 0	Predicted label 1	True label 0	470	477	True label 1	86	120
	Predicted label 0	Predicted label 1												
True label 0	470	477												
True label 1	86	120												

پیاده‌سازی با تغییر وزن به نفع برچسب با تعداد کم LLE 100:

	accuracy	recall	precision	f1	conf matrix									
rbf	0.6	0.51	0.23	0.31	 <table><tr><th>True label \ Predicted label</th><th>0</th><th>1</th></tr><tr><th>0</th><td>587</td><td>360</td></tr><tr><th>1</th><td>100</td><td>106</td></tr></table>	True label \ Predicted label	0	1	0	587	360	1	100	106
True label \ Predicted label	0	1												
0	587	360												
1	100	106												
sigmoid	0.58	0.47	0.2	0.3	 <table><tr><th>True label \ Predicted label</th><th>0</th><th>1</th></tr><tr><th>0</th><td>578</td><td>369</td></tr><tr><th>1</th><td>110</td><td>96</td></tr></table>	True label \ Predicted label	0	1	0	578	369	1	110	96
True label \ Predicted label	0	1												
0	578	369												
1	110	96												

پیاده سازی با cross validation

برای مدل با کرنل sigmoid و $c = 0.01$ و $\gamma = 0.01$

	precision	recall	f1	accuracy
k = 3	,0.48649863] ,0.50940486 [0.47266249	,0.47761509] ,0.50846749 [0.47966013	,0.20588235] ,0.17190083 [0.09854015	,0.57834461] , 0.7390625 [0.74270833
k = 5	,0.46759701] ,0.48797283 ,0.49702729 ,0.51820837 [0.53739006	,0.45621244] ,0.49297898 ,0.49814412 , 0.5126292 [0.53785247	,0.14403292] ,0.10135135 ,0.11842105 , 0.1572327 [0.23136247	,0.63920208] ,0.76909722 ,0.76736111 ,0.76736111 [0.74045139
k = 7	,0.47853481] ,0.45258169 , 0.5633265 ,0.50678137 ,0.55415174 ,0.44935047 [0.51780837	,0.47457421] ,0.43966995 ,0.52877393 ,0.50468634 , 0.5351846 ,0.47286047 [0.51545012	, 0.1483871] , 0.1097561 ,0.16494845 ,0.14096916 ,0.19909502 ,0.04878049 [0.18110236	,0.67922236] ,0.64520049 ,0.80315917 ,0.76306197 ,0.78493317 ,0.76306197 [0.7472661
k = 10	,0.49248455] ,0.46855631 ,0.45784431 , 0.59375 ,0.44645573 ,0.55012771 ,0.56460745 ,0.45817369 ,0.46755537 [0.46529909	,0.49420103] , 0.459375 , 0.465625 ,0.53541667 , 0.465625 , 0.5304113 ,0.54985688 ,0.47462497 , 0.4829757 [0.45838624	,0.13095238] ,0.13852814 ,0.08187135 , 0.171875 ,0.05228758 ,0.18543046 ,0.23809524 ,0.06622517 ,0.06993007 [0.12727273	,0.74696707] ,0.65451389 ,0.72743056 ,0.81597222 ,0.74826389 ,0.78645833 ,0.77777778 ,0.75520833 ,0.76909722 [0.66666667

پایاده‌سازی با تغییر وزن به نفع مدل با برچسب کمتر cross validation

برای مدل با کرنل sigmoid و $c = 0.01$ و $\gamma = 0.01$

	precision	recall	f1	accuracy
k = 3	,0.48503785]	,0.48740964]	,0.12692967]	,0.73503384]
	,0.51931424	,0.53320921	,0.28629579	,0.45208333
	[0.48858252	[0.48714426	[0.16068867	[0.6953125
k = 5	,0.43062771]	,0.48621846]	,0.28006088]	,0.17953166]
	,0.48670369	, 0.4775538	,0.20952381	,0.56770833
	,0.52299172	,0.54082404	,0.28848485	,0.49045139
	,0.51974657	,0.52071188	,0.20603015	,0.72569444
	[0.51127366	[0.51990145	[0.26074499	[0.55208333
k = 7	,0.50156887]	,0.50276632]	,0.24497992]	,0.54313487]
	, 0.4842462	,0.48795092	,0.11764706	,0.74483597
	, 0.4986688	,0.49775204	,0.26234568	,0.41919806
	,0.52439259	,0.54292817	,0.29139073	, 0.4799514
	,0.53469997	, 0.5620914	,0.30223881	,0.54556501
	,0.47699208	,0.47989527	,0.11764706	,0.72660996
	[0.51952257	[0.52860997	[0.28965517	[0.37424058
k = 10	,0.50624228]	,0.51107174]	,0.25555556]	, 0.5355286]
	,0.47178131	, 0.45	, 0.1954023	,0.51388889
	,0.48922559	, 0.490625	,0.13559322	, 0.734375
	,0.50347373	, 0.50625	,0.25396825	,0.51041667
	,0.53459577	,0.55520833	, 0.3035343	,0.41840278
	,0.52165218	,0.53775047	,0.28904429	,0.47048611
	,0.52986597	,0.55294535	,0.29362881	,0.55729167
	,0.49491597	,0.49479155	,0.16243655	,0.71354167
	,0.48377679	,0.47165487	,0.23887588	,0.43576389
	[0.48713324	[0.47927383	[0.20338983	[0.59201389

بررسی سوال‌های پرسیده شده :

بررسی کلی از نتایج بدست آمده :

همانطور که مشاهده می‌کنید مهمترین مشکل دادگان ما عدم تعادل بین تعداد برجسب‌های با مقدار یک و صفر بود .
این امر باعث شد همانطور که در شکل اول می‌بینید مدل با هسته rbf ما دقت ۸۳ درصدی داشته باشد اما به چه هزینه‌ای ؟
تمام دادگان را به عنوان کلاس صفر پیش‌بینی کند و از آنجایی که دادگان کلاس صفر ۸۳ درصد از کل دادگان را تشکیل می‌دادند به دقت ۸۳ درصدی برسد عملاً این یعنی اینکه مدل یاد گرفته است همه را صفر برجسب بزند و این یعنی یک مشکل بزرگ که بدی این موضوع در سه معیار دیگر به خصوص در معیار $f1$, میزان کوچکی آن به خوبی قابل تشخیص است .
لذا از نظر من مدل sigmoid با آنکه به دقت ۷۳ درصد رسید ولی به دلیل آنکه سه معیار دیگر آن از شرایط بهتری برخوردار بودند چه بسا مدل بهتری است .

یکی دیگر از عواملی که به احتمال زیاد باعث شد با در این آزمایش به دقت مناسبی نرسیم و با کاهش دادن آن مدل ما توانست به دقت بهتری برسد ابعاد مسئله یا ویژگی‌هایی استفاده شده بود .
اگر سری به دقت و معیارهای بدست آمده حاصل از کاهش ابعاد بزنید خواهیم دید که کاهش ابعاد در مدل rbf به ۳۰ واحد باعث شده است که مدل ما با این نوع کرنل به یک مدل بهتر تبدیل شود و بتواند بهتر از حالت قبل پیش‌بینی کند .

یکی دیگر از روش‌هایی که به عنوان روش پیش‌نهادهی مورد استفاده قرار گرفت در این مسئله روش **Locally Linear Embedding** در جهت کاهش ابعاد بود .
تجزیه و تحلیل اجزای اصلی (PCA) و تعبیه خطی محلی (LLE) دو روش محبوب کاهش ابعاد هستند که در تجزیه و تحلیل داده‌ها استفاده می‌شوند.

PCA یک روش خطی است که با شناسایی جهت‌های متعامد که در آن داده‌ها حداکثر تغییرات را نشان می‌دهند، داده‌های اولیه با ابعاد بالا را به فضایی با ابعاد پایین‌تر کاهش می‌دهد. این کار با محاسبه بردارهای ویژه ماتریس کوواریانس داده‌ها و انتخاب بردارهای ویژه k که اکثریت واریانس داده‌ها را توضیح می‌دهد، انجام می‌شود. از مزایای PCA می‌توان به توانایی آن در مدیریت مجموعه داده‌های بزرگ، کاهش نویز و شناسایی الگوها در داده‌ها اشاره کرد. با این حال، PCA با خطی بودن آن محدود می‌شود، زیرا فرض می‌کند که داده‌ها در یک زیر فضای خطی قرار دارند، که ممکن است همیشه درست نباشد.

از سوی دیگر، LLE یک روش غیرخطی است که ساختار هندسی محلی داده‌ها را با یافتن K -نزدیک‌ترین همسایه‌ها برای هر نقطه داده و نگاشت آنها به فضایی با ابعاد پایین‌تر از طریق یک نقشه خطی حفظ می‌کند. از مزایای LLE می‌توان به توانایی آن در مدیریت روابط غیر خطی پیچیده بین داده‌ها و توانایی آن در حفظ توپولوژی داده‌ها اشاره کرد. با این حال، LLE می‌تواند از نظر محاسباتی گران باشد و به تغییرات در تعداد همسایگان انتخاب شده حساس باشد.

بنابراین، انتخاب بین PCA و LLE به الزامات خاص آنالیز بستگی دارد. اگر هدف اصلی شناسایی اجزای اصلی داده ها و کاهش آن به یک زیرفضای خطی باشد، PCA ممکن است گزینه بهتری باشد. با این حال، اگر داده ها ساختار پیچیده و غیرخطی داشته باشند که باید حفظ شود و معاوضه در زمان محاسبه قابل قبول باشد، LLE ممکن است روش مناسب تری باشد.

حال با این تعاریف و بررسی نتایج بدست آمده از LLE می توانیم ببینیم که نسبت به PCA تاثیرات مثبت تری بر روی مدل آموزش داشته چه در حالت sigmoid، چه در حالت rbf و این نشان دهنده این است که ما یک مدل پیچیده تر سرکار داریم که احتمالا علاوه بر بالانس نبودن داده ها بین ویژگی ها نیز روابطی پیچیده تری حاکم است.

با بررسی cross validation به ازای مقادیر مختلف متوجه می شویم که تعداد cv نیز در این نوع آموزش یک هاپیر پارامتر مهم است که در این آزمایش به ازای $K = 5$ قابل قبول ترین نتیجه را گرفتیم.

اما مهمترین بخش این قسمت از پروژه پاسخ دادن به سوال زیر است که به خوبی به کمک این نوع دادگان آن را تجربه کردم.

- آیا اگر مجموعه دادگان دارای عدم تعادل باشد (یعنی از هر کلاس به تعداد تقریبا یکسان داده موجود نباشد، برای یک کلاس تعدادی خیلی کم و برای دیگری تعدادی بسیار و یا تا حدی بیشتر)، آیا رده بندی ماشین بردار پشتیبان دچار مشکل می شود و رده بندی توسط آن تحت الشعاع قرار می گیرد؟ چرا؟ به طور کامل توضیح دهید.

- راه حل مشکل بالا یعنی عدم تعادل مجموعه دادگان چیست و تاثیرات کلی آن را روی عملیات رده بندی توسط هر رده بندی را توضیح دهید.

بله و بسیار بد هم ممکن است تحت تاثیر قرار بگیرد برای مثال بر روی همین دادگان مدل rbf اصلا سعی نمی کرد که اصلا برچسب های یک را یاد بگیرد و حتی اگر هم به ازای مقادیر بالای c یاد میگرفت کامل بر روی آن برچسب ها با تعداد کم overfit می شد و این یعنی یادگیری نادرست.

یکی از راه حل های حل مشکل بالا دادن وزن به برچسب ها است یعنی به برچسب با فراوانی کم وزن بیشتری بدهیم که هنگام محاسبه خطا تا جایی که می تواند سعی کند به گونه ای یاد بگیرد که دادگان با برچسب کم جز دادگان خطا نباشد که خود پیدا کردن وزن مناسب نیز یک چالش بزرگ است. برای حل این مشکل قسمتی به نام تغییر وزن را اضافه کردم و نتایج آن را نیز در کنار نتایج بدون وزن آوردم.

همانطور که می بینید مقدار دقت کاهش زیادی داشته ولی به همان میزان میزان $f1$ بالا آمده و ماشین حال دو برچسب را در یک حالت برابر تر یاد می گیرد اما خوب از هر دو برچسب با خطا یاد می گیرد .

به اصطلاح یک مشکل یعنی عدم یادگیری برچسب یک را حل کردیم اما به دلیل اینکه احتمالا ویژگی های نزدیک به هم دیگر داشتند برچسب های شماره یک و صفر در نتیجه مدل ما با قرار دادن مقادیر زیادی از دادگان برچسب صفر در کنار برچسب یک به عنوان یک گروه باعث شده است که عملا دقت ما کاهش پیدا کند .

که این نشان می دهد که غیر از اینکه مدل بالانس نیست پیچیده هم هست البته با کاهش ابعاد به روش های یاد شده به دقت بالاتری دست پیدا کردیم در مدل rbf که این نشان می دهد که گاهی کاهش ابعاد چقدر به ما کمک می کند.

اما با این وجود آیا این دادگان به خوبی توسط ماشین یادگیری SVC یاد گرفته شدند و به دقت قابل قبولی رسیدند ؟

جواب ساده است .

خیر

عملا در اینجا یک درس بزرگتر نیز گرفتیم که گاهی بعضی از ماشین ها یادگیری برای بعضی از مدل دادگان خاص به خوبی کار نمی کنند و نتیجه مورد علاقه ما را بر نمی گردانند هرچند ما در این آزمایش سعی کردیم تمامی تلاش خود را برای پیدا کردن مدل با پارامترهای بهینه را پیدا کنیم اما به احتمال زیاد استفاده از این ماشین را به کسی پیشنهاد نمی کنیم .

آیا راهی هست که بتوان از svm برای یادگیری این دادگان استفاده کرد .

بله!

اما نیازمند preprocess قوی تر هست که علاوه بر آنکه به خوبی معرف کل جامعیت دادگان هست بتوان از آن به خوبی برای حل مسئله به کمک SVM نیز استفاده کرد .

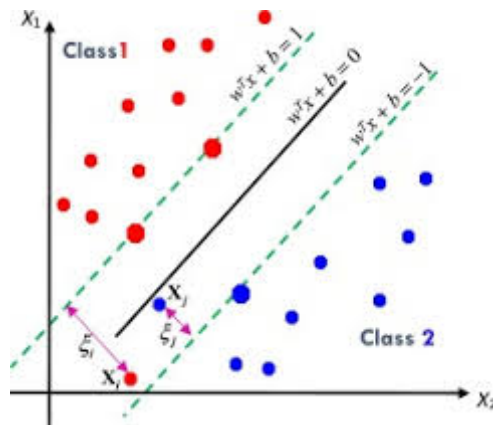
حقیقت آن است که ما در این آزمایش سعی کردیم یک سری از ستون ها را که کار را برای پیش بینی آسان می کرد را حذف کنیم همچنین سعی کردیم تعداد فیچرهای بیشتری از داده استخراج کنیم تا بتوانیم با مدل $X Boost$ که بروی همین دادگان کار کرده بود و نتیجه قابل قبولی گرفته بود ماشین خود را مقایسه کنیم ایجاد چالش بیشتر در جهت یادگیری و فهم یک سری مسائل که صرفا هنگام کار با دادگان واقعی با آن روبه رو خواهیم شد .

در مورد متغیر های slack به سوالات زیر پاسخ دهید:

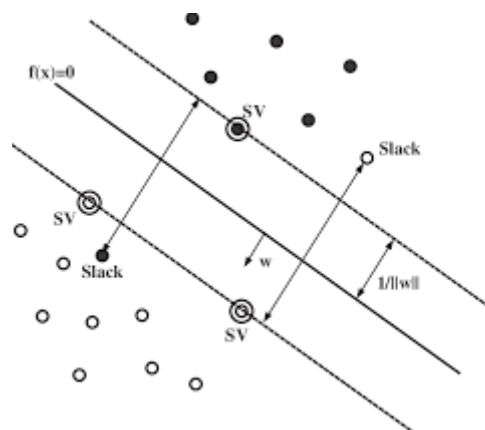
اگر مقدار این متغیر بین 0 و 1 باشد به چه معناست؟ اگر بزرگ تر از 1 باشد چطور؟

در طبقه بندی ماشین بردار پشتیبان (SVM)، متغیر slack یک متغیر با ارزش واقعی غیر منفی است که نشان دهنده میزان نقض (به عنوان مثال، طبقه بندی اشتباه) محدودیت های حاشیه برای یک نقطه داده معین است.

اگر مقدار متغیر slack بین 0 و 1 باشد، به این معنی است که نقطه داده مربوطه در حاشیه کلاس صحیح اما در سمت اشتباه مرز تصمیم قرار دارد. چنین نقاط داده ای "SOFT" یا "Margin errors" نامیده می شوند زیرا اجازه دارند محدودیت حاشیه را تا حدی نقض کنند. در شکل زیر آن نقطه آبی رنگ که به نوار اصلی نزدیک است در این بازه قرار می گیرد.



از طرف دیگر، اگر مقدار متغیر slack بزرگتر از 1 باشد، به این معنی است که نقطه داده مربوطه در سمت اشتباه مرز تصمیم و خارج از حاشیه کلاس صحیح قرار دارد. این نقاط داده، خطاهای حاشیه "Hard" نامیده می شوند و مجاز به نقض محدودیت های حاشیه نیستند. مقادیر زیاد متغیر slack ممکن است نشان دهنده این باشد که مدل SVM قادر به تفکیک داده ها به طور موثر نیست یا موارد پرت در داده ها وجود دارد در شکل زیر نمونه ای این نوع slack آورده شده است.



با توجه به آنچه گفته شد، تعداد متغیرهای Slack در SVM با میزان سوگیری و واریانس مرتبط است. تعداد بیشتر متغیرهای Slack به این معنی است که ما به مدل انعطاف پذیری بیشتری می دهیم و احتمالاً مدل دارای سوگیری کمتر اما واریانس بیشتری است. این به این دلیل است که نسبت به طبقه بندی های نادرست تحمل بیشتری دارد.

در مورد حاشیه خط تقسیم، تعداد متغیرهای slack با آن رابطه معکوس دارد. متغیرهای Slack با فاصله آنها تا حاشیه جریمه می شوند، بنابراین یک ثابت بزرگ اشتقاق از حاشیه را جریمه می کند. بنابراین داشتن متغیرهای Slack بیشتر به این معنی است که به نقاط اجازه می دهیم خط حاشیه اصلی را بشکنند و از این رو حاشیه را گسترده تر کنیم. اگر متغیرهای slack کمتری داشته باشیم، در این صورت قانون طبقه بندی سخت تری را اعمال می کنیم و حاشیه کوچکتر خواهد بود.

افزایش تعداد متغیرهای Slack منجر به انعطاف پذیری بیشتر در طبقه بندی کننده می شود و می تواند منجر به تعداد بردارهای پشتیبان کمتر شود. برعکس، کاهش تعداد متغیرهای Slack می تواند تعداد بردارهای پشتیبان را افزایش دهد. با این حال، تعداد بردارهای پشتیبان نیز تحت تأثیر عوامل دیگری مانند پیچیدگی داده ها و نوع عملکرد هسته مورد استفاده قرار می گیرد. باییم و آزمایش زیر را برای بهترین مدل انجام دهیم :

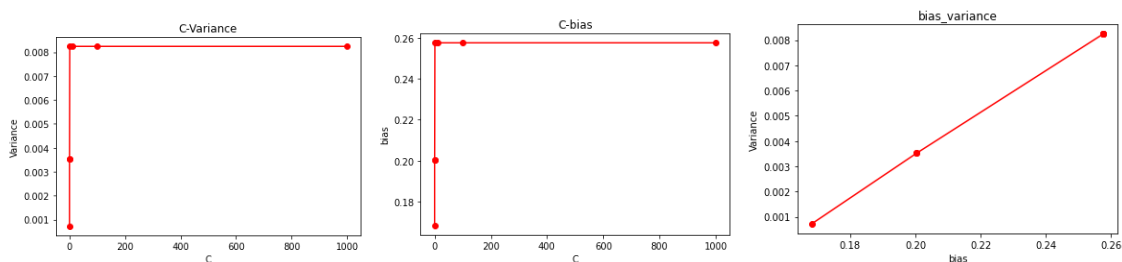
برای پارامتر C در نظر بگیرید و مقادیر بایاس و واریانس را در هر حالت چاپ کنید و نتایج را در گزارش خود تحلیل و بررسی کنید.

حاصل برابر خواهد بود با :

```

C = 0.001, f1_score = 0.000 accuracy = 0.832, bias = 0.168, variance = 0.001
C = 0.01, f1_score = 0.049 accuracy = 0.800, bias = 0.200, variance = 0.004
C = 0.1, f1_score = 0.049 accuracy = 0.800, bias = 0.200, variance = 0.004
C = 1.0, f1_score = 0.124 accuracy = 0.742, bias = 0.258, variance = 0.008
C = 10.0, f1_score = 0.124 accuracy = 0.742, bias = 0.258, variance = 0.008
C = 100.0, f1_score = 0.124 accuracy = 0.742, bias = 0.258, variance = 0.008
C = 1000.0, f1_score = 0.124 accuracy = 0.742, bias = 0.258, variance = 0.008

```



این هم نمودارهای مختلف آن نسبت به یک دیگر سمت راستی bias به عنوان X و واریانس به عنوان Y است .

وسطی مقادیر مختلف بایاس را به ازای C های مختلف می بینیم .

سمت چپ نیز مقادیر مختلف واریانس را به ازای C های مختلف می بینیم .

همانطور که می بینیم برای این مدل از دادگان با افزایش C واریانس افزایش پیدا کرده و در کنار آن بایاس نیز افزایش یافته است .

افزایش واریانس به معنی آن است که قدرت عمومیت مدل ما کاهش پیدا کرده است که نتیجه ای درستی است و افزایش بایاس نیز نشان می دهد سوگیری ما افزایش پیدا کرده است یعنی خطا افزایش پیدا کرده است .

یعنی با کاهش قدرت عمومیت مدل خطا نیز افزایش پیدا کرده است .

بخش سوم)

ابتدا تابع زیر را ساختم که به کمک آن نقاط را تولید کنم. ب.
برای حالت پیچیده :

```
import math

def function_rez_complex(x, y):
    """
    This function takes in three arguments, x, y, and returns the result of x cubed plus the sine of y.
    """
    return x**3 + math.sin(y)
```

برای حالت ساده :

```
def function_rez_simple(x,y):

    return x**2+3*y
```

سپس به صورت رندوم به این توابع نقاطی از x, y را می‌دهم و خروجی را میگیرم.

```
import numpy as np

# set the seed for reproducibility
np.random.seed(42)

# create a random numpy array with 2 | features and 5000 rows
array_3x5000 = np.random.randint(-100, 101, size=(5000, 2))

function_res_simple = np.array([function_rez_simple(item[0],item[1]) for item in array_3x5000])
print(function_res_simple)
```

همانند مسئله قبل نیز یک تابع **experiment model** می‌سازیم ولی به جای SVC از SVR استفاده می‌کنیم .

```
def experiment_model(X_train,y_train,X_test):

    result_list = []

    # Initialize the SVM with a rbf kernel
    svm_model = SVR(kernel='rbf')

    # Define the parameter grid for gamma and C
    param_grid = {"gamma": ["auto", "scale"], 'C': [0.0001, 0.001, 0.1, 1, 10, 100, 1000]}
    clf_one = GridSearchCV(estimator=svm_model, param_grid=param_grid)
    clf_one.fit(X_train, y_train)
    Predicted_labels= clf_one.predict(X_test)
    result_list.append((Predicted_labels,clf_one))
    print(f"SVM , Kernel = RBF , Best parameters: {clf_one.best_params_}")

    print("SVM , Kernel = Linear ")
    param_grid = {'C': [0.0001, 0.001, 0.1, 1, 10, 100, 1000]}
    svm_model = LinearSVR()
    clf_two = GridSearchCV(estimator=svm_model, param_grid=param_grid)
    clf_two.fit(X_train, y_train)
    Predicted_labels= clf_two.predict(X_test)
    result_list.append((Predicted_labels,clf_two))
    print(f"SVM , Kernel = Linear , Best parameters: {clf_two.best_params_}")

    #Initialize the SVM with a polynomial kernel
    param_grid = {"degree": [2, 3, 4], "C": [0.0001, 0.001, 0.1, 1, 10, 100, 1000]}
    svm_model = SVR(kernel='poly')
    clf_three = GridSearchCV(estimator=svm_model, param_grid=param_grid,cv=5)
    clf_three.fit(X_train, y_train)
    Predicted_labels= clf_three.predict(X_test)
    result_list.append((Predicted_labels,clf_three))
    print(f"SVM , Kernel = Poly , Best parameters: {clf_three.best_params_}")

    # Initialize the SVM with a Sigmoid kernel
    param_grid = {'gamma': [0.1, 1, 10], 'C': [0.0001, 0.001, 0.1, 1, 10, 100, 1000]}
    svm_model = SVR(kernel='sigmoid')
    clf_four = GridSearchCV(estimator=svm_model, param_grid=param_grid,cv=5)
    clf_four.fit(X_train, y_train)
    Predicted_labels= clf_four.predict(X_test)
    result_list.append((Predicted_labels,clf_four))
    print(f"SVM , Kernel = Sigmoid , Best parameters: {clf_four.best_params_}")

    return result_list
```

نتایج برای شکل ساده :

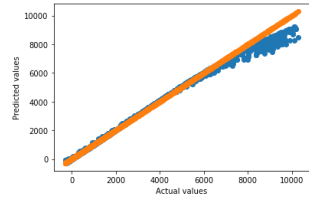
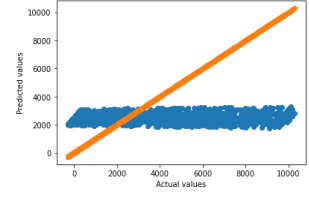
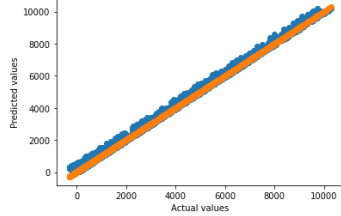
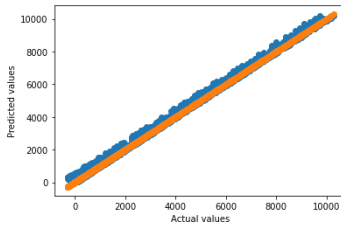
بهترین پارامترها برای هر هسته خواهد بود :

'SVM , Kernel = RBF , Best parameters: 'C': 1000, 'gamma': 'scale

SVM , Kernel = Linear , Best parameters: 'C': 100

SVM , Kernel = Poly , Best parameters:'C': 100, 'degree': 2

SVM , Kernel = Sigmoid , Best parameters:'C': 1, 'gamma': 0.1

	score	MSE	RMSE	MAE	
RBF	0.9	2758	52	20	
Linear	0.09-	9844170	.3137	2533	
Poly	0.99	46760	216	149	
Sigmoid	0.07-	9664434	3108	2523	

شکل نشان می‌دهد چقدر پیش‌بینی‌ها بهم نزدیک هستند. (محور x نشان دهنده مقدار واقعی و محور y نشان دهنده مقدار پیش‌بینی هست)

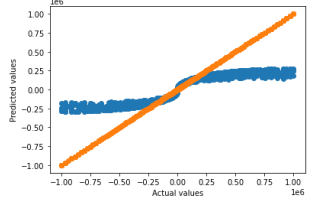
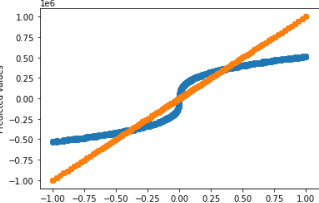
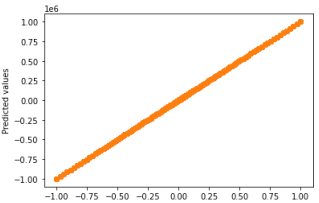
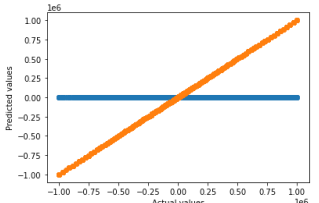
نتایج برای شکل پیچیده تر:

SVM , Kernel = RBF , Best parameters: 'C': 1000, 'gamma': 'scale

SVM , Kernel = Linear , Best parameters: 'C': 100

SVM , Kernel = Poly , Best parameters: 'C': 1000, 'degree': 3

SVM , Kernel = Sigmoid , Best parameters: 'C': 100, 'gamma': 0.1

	score	MSE	RMSE	MAE	
RBF	0.56	63904168535	252792	153252	
Linear	0.82	25625128731	160078	125780	
Poly	0.99	0.66	0.81	0.7	
Sigmoid	0.30	101813113139	319081	204222	

شکل نشان می دهد چقدر پیش بینی ها بهم نزدیک هستند. (محور x نشان دهنده مقدار واقعی و محور y نشان دهنده مقدار پیش بینی هست)

همانطور که انتظار داشتیم تابع هسته با کرنل poly روی مدل‌های من بهتر جواب داد علت آنکه توابع که چه برای ساده انتخاب کردم و چه برای پیچیده تر می‌توانی در آنان شاخصه مهمتری برای تغییر بود و همانطور هم که دیدیم مدل هسته مختص این نوع دادگان یعنی poly بهتر جواب داد و به درستی و با دقت کامل درجه توانی خود را به مدل موردنظر هم در توان ۲ و هم در توان ۳ درست ست کرد .

هرچند که مدل ما وقتی پیچیده تر شد کرنل‌های دیگر که شباهت کمتری به فرمول اصلی داشتند در جایگاه بسیار بدتری قرار گرفتند . برای مثال مدل rbf در مدل ساده تر توانست خود را به گونه‌ای تطبیق دهد اما با پیچیده کردن آن تنها مدل poly بود که توانست بهترین پیش‌بینی را از مدل داشته باشد .

بخش چهارم)

در بخش چهارم به آموزش یک تابع رگرسیون را برای پیش‌بینی تعداد افرادی که در هر روز دوچرخه را به اشتراک می‌گذارند می‌پردازیم .

خوشبختانه این دیتاست missing دیتا نداشت و صرفاً سطر مربوط به تاریخ را کنار می‌گذاریم در این آزمایش .

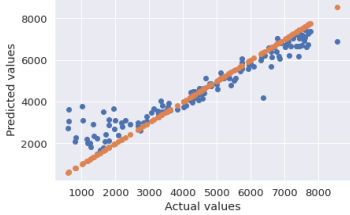
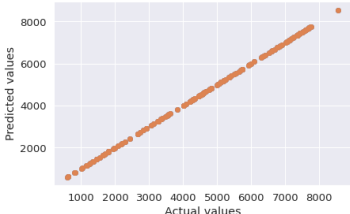
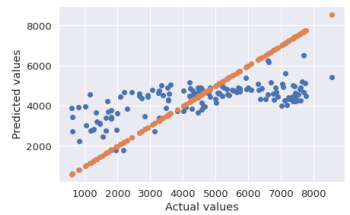
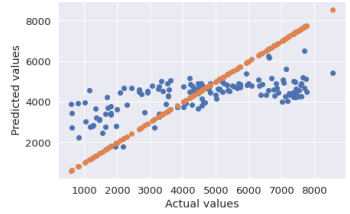
سپس داده‌ها را نرمالیز می‌کنیم و با تقسیم بندی دادگان آموزش و تست می‌رویم همانند سراغ کار بالا دنبال مدل با بهترین پارامتر و هسته برای این دادگان می‌گردیم .

'SVM , Kernel = RBF , Best parameters: 'C': 1000, 'gamma': 'auto

SVM , Kernel = Linear , Best parameters: 'C': 1000

SVM , Kernel = Poly , Best parameters: 'C': 1000, 'degree': 3

SVM , Kernel = Sigmoid , Best parameters: 'C': 100, 'gamma': 0.1

	score	MSE	RMSE	MAE	
RBF	0.96	153903	392	186	
Linear	1	9.7	0.01	0.007	
Poly	0.89	447328	668	362	
Sigmoid	0.92	342083	584	404	

مدل linear برخلاف انتظار بهترین نتیجه را در این آزمایش گرفت و توانست این مسئله regression را به خوبی تقریب بزند و در جایی که مدل با کرنل های پیچیده تر نتایج غیر قابل قبولی از خود نشان دادند و صرفا مدل rbf توانست مدل بهتری را ارائه دهد . با این وجود مدل linear به ازای $c = 1000$ به این دقت رسیده است که نشان می دهد که به احتمال زیاد بر روی دادگان آموزش به شکل قابل ملاحظه ای fit شده است ولی خوب توانسته با همین قدرت انعطاف پذیری کم دقت خوبی بر روی دادگان تست بدست آورد که نشان از این دارد که براساس دادگانی که داریم یک فرمولیشن خطی بین دادگان برقرار است که با یادگیری آن به خوبی می توان خروجی را پیش بینی کرد .