למידה סטטיסטית מבוססת נתונים - חורף - 096411

HW4

מגישים:

איתי ברקוביץ 039632732

אילן פרנק 043493386

$\sum x_i = 0$, $y_i = w_0 + x_i w_1 + \varepsilon$

$$l = \arg\min \left\{ (wx - y)^2 + \lambda w^2 \right\}$$

$$\frac{\partial l}{\partial w_0} = 2(wx - y) = 0 \rightarrow n w_0 + w_1 \sum x_i - \sum y = 0$$

$$\rightarrow \boxed{\hat{w}_0 = \bar{y}}$$

$$\frac{\partial l}{\partial w_1} = 2(wx - y) x_i + 2\lambda w_1 = 0$$

$$\Rightarrow (\bar{y} + w_1 x_i - y_i) x_i + \lambda w_1 = 0$$

$$\Rightarrow \bar{y} \sum x_i + w_1 \sum x_i^2 - \sum y_i x_i + \lambda w_1$$

$$\Rightarrow \boxed{\hat{w}_1 = \frac{\sum y_i x_i}{\sum x_i^2 + \lambda}}$$

סעיף ג:

- נחזור לרגרסיה לינארית פשוטה, נחפש מינימום לביטוי הבא:

$$\arg\min \sum (w_0 + w_1 x_i - y_i)^2$$ מינימום

- נאמר בשאלה שאין ... מכיוון ש $w_0 = 0$ נכתוב את הביטוי:

$$= \arg\min \sum (w_1 x_i - y_i)^2$$ מינימום

- נגזור לפי w ונשווה לאפס למציאת מינימום:

$$\frac{\partial f}{\partial w_1} = 2\sum (w x_i - y_i)\cdot x_i = 0$$

$$\Rightarrow w_1 X = y \rightarrow \boxed{w_1 X \cdot X^T = X^T y}$$ בצורה מטריצית

- נמצא את התוחלת של y

לפי ההגדרה: $y = wx + \varepsilon$

$$E(y) = E(wx + \varepsilon) = E(wx) + E(\varepsilon) = E(wx) = X \cdot E(w) = X \cdot w = w \cdot x$$

0 לפי הנתון; לפי הגדרה

- כעת נחשב תוחלת ל- $\hat{w}$

$$E(\hat{w}_1) = E\left(\frac{X^T y}{X^T X}\right) = \frac{X^T E(y)}{X \cdot X^T} = \frac{X^T \cdot X \cdot w_1}{X^T X}$$

- נחשב שונות של w (ללא רגולריזציה)

$$Var(w_1) = E(w_1 - E(w_1))^2 = E\left(\frac{X^T y}{X \cdot X^T} - w_1\right)^2 = E\left[\left(\frac{X^T y}{X \cdot X}\right)^2\right] - E^2(w_1)$$

$$= \left(\frac{X^T}{X \cdot X^T}\right)^2 E(y^2) - w_1^2 = \frac{X^T X}{(X \cdot X^T)^2} E(w_1^2 x^2 + \varepsilon w x + \varepsilon^2) - w^2$$

$$= w^2 + \overset{0}{E(\varepsilon)}\, E(wx) \frac{X^T X}{(X \cdot X^T)^2} + \frac{\overset{\sigma}{E(\varepsilon^2)}}{X \cdot X^T} - w^2 = \frac{\sigma}{X^2}$$

כעת נחשב שונות עבור $\hat{w}_1$

$$Var(\hat{w}_1) = Var\left(\frac{X^TX}{X^TX+\lambda} \cdot w_1\right) = \left(\frac{X^TX}{X^TX+\lambda}\right)^2 Var(w_1) = \frac{(X^TX)^2}{(X^TX+\lambda)^2} \cdot \frac{\sigma}{(X^TX)}$$

$$\boxed{Var(\hat{w}_1) = \frac{X^TX\sigma}{(X^TX+\lambda)^2}}$$

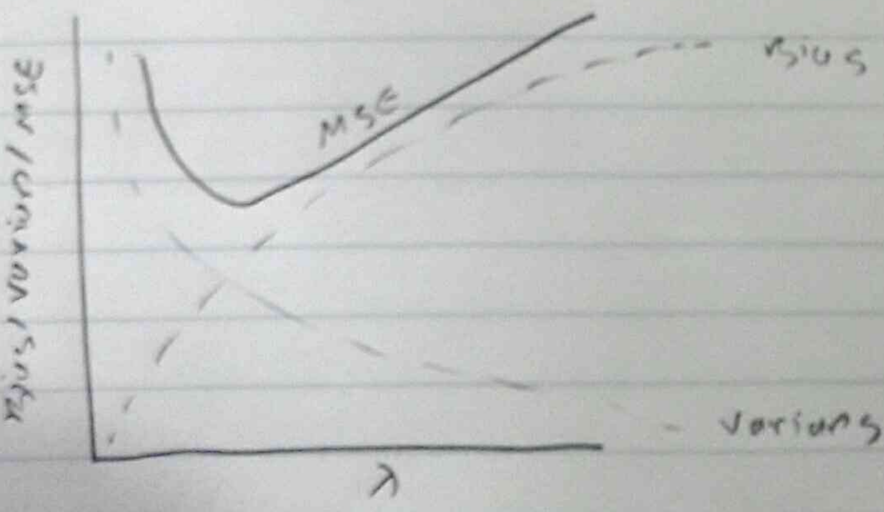
סעיף ג:

• ככל הגדלת למדא מוביל לנו את הקטנת ה- bias
ל- variance:

$$MSE(\hat{w}) = E\left[(\hat{w}_1 - w_1)^2\right] = Var(\hat{w}_1) + bias(\hat{w}_1)^2$$

$$= \frac{X^TX\sigma}{(X^TX+\lambda)^2} + \left(w_1 \cdot \frac{X^TX}{X^TX+\lambda}\right)$$

λ ↑ - ביאס עולה לנו את המשקל הגדל עבור הלמדא
λ ↑ - variance יורד לנו את המשקל הגדל לערך המינימלי
התרגום MSE נראה כך:



• ככל שנגדיל את λ, ה-w יהיו יותר קטנים ואף לא
שואפים לאפסים אך שונים מ- ω

② ב. אומד הריבועים הפחותים: $(\hat{w}_1, \hat{w}_2) = \arg\min_{w_1,w_2} \sum (y_i - w_1 - w_2 x_i)^2$

נסמן (כמו בהמשך התרגיל) $x_2 = -x_1$, $y_2 = -y_1$

נגזור למציאת המינימום!

ונשווה ל-0

$$\frac{\partial}{\partial w_1} = -2 \sum (y_i - w_1 - w_2 x_i) = 0$$

$$= y_1 - y_1 - 2w_1 - w_2(x_1 - x_1) = -2w_1 = 0 \Rightarrow \hat{w}_1 = 0$$

$$\frac{\partial}{\partial w_2} = -2 \sum (y_i - w_1 - w_2 x_i) x_i = 0$$

$$= -\underbrace{w_1 \sum x_i}_{0} + \sum y_i x_i - w_2 \sum x_i^2 = 0$$

$$w_2 = \frac{\sum y_i x_i}{\sum x_i^2} = \frac{x_1 y_1 + (-x_1)(-y_1)}{x_1^2 + (-x_1)^2} = \frac{2 x_1 y_1}{2 x_1^2} = \frac{y_1}{x_1}$$

ג. אומד ridge: $(\hat{w}_1, \hat{w}_2) = \arg\min_{w_1,w_2} \lambda (w_1^2 + w_2^2) + \sum (y_i - w_1 - w_2 x_i)^2$

נגזור ונשווה ל-0 כדי למצוא מינימום!

$$\frac{\partial}{\partial w_1} = 2\lambda w_1 - 2 \sum (y_i - w_1 - w_2 x_i) = 0$$

$$\lambda w_1 + 2 w_1 - \sum y_i + w_2 \sum x_i = 0 \qquad \sum x_i = \sum y_i = 0$$

$$\Rightarrow w_1(\lambda + 2) = 0 \qquad 0 < \lambda \Rightarrow \hat{w}_1 = 0$$

$$\frac{\partial}{\partial w_2} = 2\lambda w_2 - 2 \sum (y_i - w_1 - w_2 x_i) x_i = 0$$

$$\Rightarrow 2\lambda w_2 - 2 \sum x_i y_i + 2 w_1 \sum x_i + 2 w_2 \sum x_i^2 = 0$$

$$w_2 (\lambda + \sum x_i^2) \overset{0}{=} \sum x_i y_i$$

$$\hat{w}_2 = \frac{\sum x_i y_i}{\lambda + \sum x_i^2} = \frac{2 x_1 y_1}{\lambda + 2 x_1^2} = \frac{x_1 y_1}{\frac{\lambda}{2} + x_1^2}$$

d. אומד Lasso: $\hat{w}_1, \hat{w}_2 = \arg\min \lambda |w_1| + \lambda |w_2| + \sum (y_i - w_1 - w_2 x_i)^2$

בגלל הערך המוחלט לא ניתן לגזור. נפתח את הביטוי:

$$= \arg\min \lambda |w_1| + \lambda |w_2| + \underbrace{((y_1 - w_2 x_1) - w_1)^2}_{(a-b)^2} + \underbrace{((-y_1 + w_2 x_1) - w_1)^2}_{(-a-b)^2 = (a+b)^2}$$

$$\arg\min \lambda |w_1| + \lambda |w_2| + 2 (y_1 - w_2 x_1)^2 + 2 w_1^2 \qquad \Leftarrow (a-b)^2 + (a+b)^2 = 2a^2 + 2b^2$$

המחוברים השני והשלישי לא תלויים ב-w_1, ונוכל במינימום החשבון ורכיבי

מינימליים כאשר $|w_1|$-ש כך מינימלי, לכן $\hat{w}_1$ מתקבל ערכו $w_1 = 0$

מניין שראינו בסעיף 4,1 אינה תלויה ב-w_2, $\hat{w}_2 = \arg\min_{w} \lambda |w_2| + 2 (y_1 - w_2 x_1)^2$

3. רצינו כי $\hat{w}_2^L = \arg\min_{w_2} \lambda|w_2| + 2(y_1 - w_2x_1)^2$

הבעיה אינה גזירה בנקודה $w_2=0$, אולם בשאר הנקודות היא גזירה.

נחלק למקרים $0<w_2$, $w_2<0$ ונגזור כל אחד מהם:

$w_2>0$: $\frac{\partial}{\partial w_2} = \lambda - 4x_1(y_1 - w_2x_1) = \lambda - 4x_1y_1 + 4x_1^2w_2 > 0$

מכיוון ש $\lambda > 4|x_1y_1|$, $[\lambda - 4x_1y_1 \geq \lambda - 4|x_1y_1| > 0]$, $0 < 4x_1^2w_2$ שכן $0<w_2$, $0<x_1^2$

לכן הפונקציה עולה בתחום $(0,\infty)$

$w_2<0$: $-\lambda - 4x_1y_1 + 4x_1^2w_2 < 0$

נפתח את הביטוי ונחלק למקרים

$$\arg\min_{w_2} \lambda|w_2| + 2y_1^2 - 4x_1y_1w_2 + 2x_1^2w_2^2$$

$$\arg\min \begin{cases} \arg\min 2x_1^2w_2^2 + (\lambda - 4x_1y_1)w_2 + 2y_1^2, & \text{עבור } 0<w_2 \\ \arg\min_{w_2} 2x_1^2w_2^2 + (-\lambda - 4x_1y_1)w_2 + 2y_1^2 & \text{עבור } w_2<0 \end{cases}$$

עבור $0<w_2$ נקבל

$$\lambda - 4x_1y_1 \geq \lambda - 4|x_1y_1| > 0 \Rightarrow (\lambda - 4x_1y_1)w_2 > 0$$

בכל התחום, וכמו כן, $0 < 2x_1^2w_2^2$, וגם.

עבור $w_2<0$ נקבל

$$-\lambda - 4x_1y_1 = -(\lambda + 4x_1y_1) < 0 \Rightarrow (-\lambda - 4x_1y_1)w_2 > 0$$

וכך גם $2x_1^2w_2^2$

לכן בהכרח ערכי הפונקציה עבור $w_2 \neq 0$ $>$ $2y_1^2$, שכן הערך מתקבל עבור $w_2=0$, והוא המינימום של הפונקציה, לכן $\hat{w}_2 = 0$

שאלה 3

א) $w_0=0.1,\ w_1=-0.1,\ w_2=0.3$

סעיף א':

$$h(x)=P(y=1\mid X=x)=\frac{1}{1+e^{-\langle w x\rangle}}$$

i	$\langle w x\rangle$	$h(x)$
1	-0.3	0.42
2	0.8	0.68
3	-1.1	0.25
4	0.2	0.54
5	-0.4	0.40

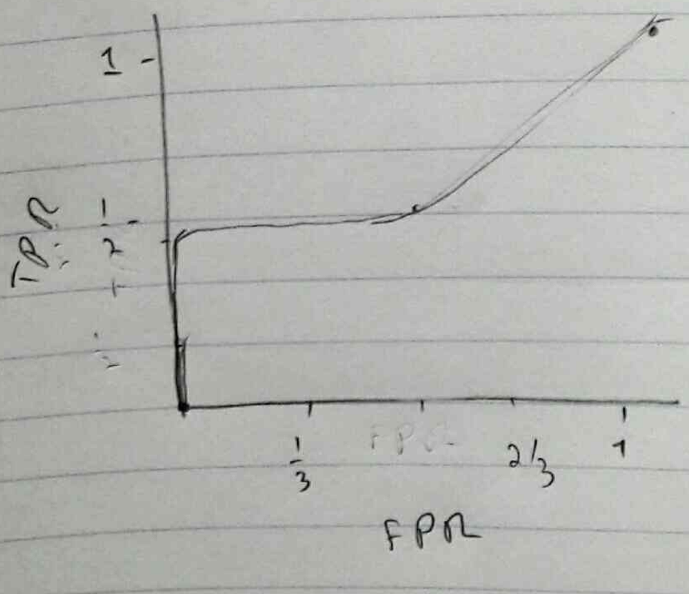
* עבור רגרסיה לוגיסטית כאשר

→ Predic = 1 ⟸ $h(x)\geq 0.5$

→ Predic = 0 ⟸ $h(x)<0.5$

סעיף ב':

True label	Predic	Predic	TPR	FPR
0	0.25 → 0 ↑0	0.26	1	2/3
1	0.4 → 0 ↓+1	0.41	1/2	2/3
0	0.42 → 0 ↓	0.5	1/2	1/3
0	0.54 ↓	0.55	1/2	0
1	0.68			



In [1]:

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

In [2]:

```
def adjust_labels_to_binary(y_train, target_class_value):

    target = None

    if target_class_value=='Setosa':
        target = 0

    if target_class_value=='Versicolour':
        target = 1

    if target_class_value=='Virginicacv':
        target = 2

    ans = (y == target).astype(int) - (y != target).astype(int)
    return ans.astype(int)
```

In [3]:

```
def one_vs_rest(x_train, y_train, target_class_value):
    y_train_binarized = adjust_labels_to_binary(y_train, target_class_value)
    model = LogisticRegression()
    return model.fit(x_train,y_train_binarized)
```

In [34]:

```
def binarized_confusion_matrix(X, y_binarized, one_vs_rest_model, prob_threshold):
    predic = np.array(one_vs_rest_model.predict_proba(X))
    predic_threshold_prob = np.zeros(len(predic))

    for i in range(len(predic)):
        if predic[i,1] > prob_threshold:
            predic_threshold_prob[i] = 1
        else:
            predic_threshold_prob[i] = -1

    temp = predic_threshold_prob-y_binarized
    tp = sum([1 if predic_threshold_prob[i]==y_binarized[i]==1 else 0 for i in range(le
n(y_binarized))])
    tn = sum([1 if predic_threshold_prob[i]==y_binarized[i]==-1 else 0 for i in range(l
en(y_binarized))])
    fp = sum([1 if predic_threshold_prob[i]==1 and y_binarized[i]==-1 else 0 for i in r
ange(len(y_binarized))])
    fn = sum([1 if predic_threshold_prob[i]==-1 and y_binarized[i]==1 else 0 for i in r
ange(len(y_binarized))])
    return np.array([[tp,fn],[fp,tn]])
```

In [7]:

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

In [8]:

```
prob_threshold = 0.5
```

In [9]:

```
df = pd.DataFrame(0, index=range(len(y)), columns=range(3))
```

In [37]:

```
#7 סעיף

for i in ('Setosa','Versicolour','Virginicacv'):
    y_binarized = adjust_labels_to_binary(y,i)
    one_vs_rest_model = one_vs_rest(X, y_binarized,i)
    c_matrix = binarized_confusion_matrix(X, y_binarized, one_vs_rest_model, prob_thres
hold)
    print (c_matrix)
```

```
[[ 50   0]
 [  0 100]]
[[15 35]
 [10 90]]
[[50  0]
 [ 3 97]]
```

In [38]:

```
all_target_class_dict = {'Setosa': one_vs_rest(X, y,'Setosa'),
                         'Versicolour': one_vs_rest(X, y,'Versicolour'),
                         'Virginicacv': one_vs_rest(X, y,'Virginicacv')  }
```

In [39]:

```
#סעיף ד
def micro_avg_precision(X, y, all_target_class_dict, prob_threshold):
    tp = 0
    fp = 0

    for i in all_target_class_dict:
        y_binarized = adjust_labels_to_binary(y,i)
        c_matrix = binarized_confusion_matrix(X, y_binarized, all_target_class_dict[i],
 prob_threshold)
        tp = tp + c_matrix[0,0]
        fp = fp + c_matrix[1,0]

    return tp/(tp+fp)
```

In [40]:

```
micro_avg_precision(X, y, all_target_class_dict, prob_threshold)
```

Out[40]:

```
0.8984375
```

In [41]:

```
#סעיף ו

def micro_avg_recall(X, y, all_target_class_dict, prob_threshold):
    tp = 0
    fn = 0

    for i in all_target_class_dict:
        y_binarized = adjust_labels_to_binary(y,i)
        c_matrix = binarized_confusion_matrix(X, y_binarized, all_target_class_dict[i],
 prob_threshold)
        tp = tp + c_matrix[0, 0]
        fn = fn + c_matrix[0, 1]

    return tp/(tp+fn)
```

In [42]:

```
micro_avg_recall(X, y, all_target_class_dict, prob_threshold)
```

Out[42]:

```
0.76666666666666672
```

In [43]:

```
#סעיף ז
def micro_avg_false_positve_rate(X, y, all_target_class_dict, prob_threshold):
    fp = 0
    tn = 0
    for i in all_target_class_dict:
        y_binarized = adjust_labels_to_binary(y,i)
        c_matrix = binarized_confusion_matrix(X, y_binarized, all_target_class_dict[i],
 prob_threshold)
        fp = fp + c_matrix[1, 0]
        tn = tn + c_matrix[1, 1]
    return fp/(tn+fp)
```

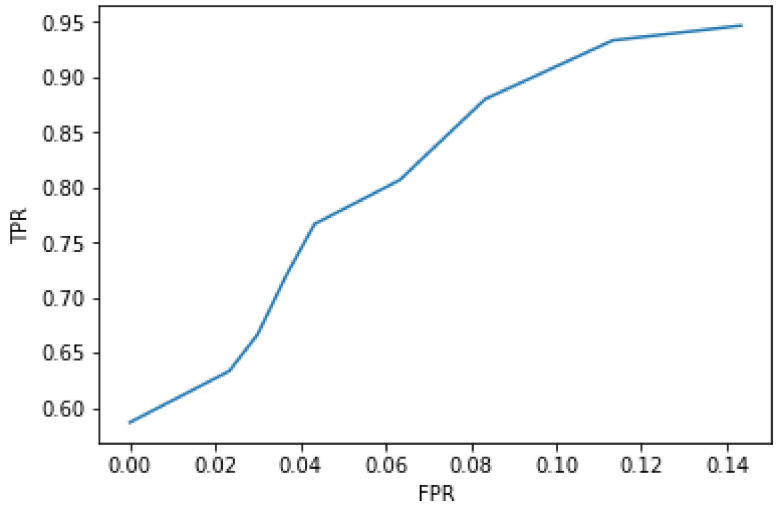
In [44]:

```
#סעיף ח
threshold = [0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75]
tpr = np.zeros(len(threshold))
fpr = np.zeros(len(threshold))
for i in range(len(threshold)):
    tpr[i]=micro_avg_recall(X, y, all_target_class_dict, threshold[i])
    fpr[i] = micro_avg_false_positve_rate(X, y, all_target_class_dict, threshold[i])

plt.figure()
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
```

Out[44]:

```
Text(0,0.5,'TPR')
```

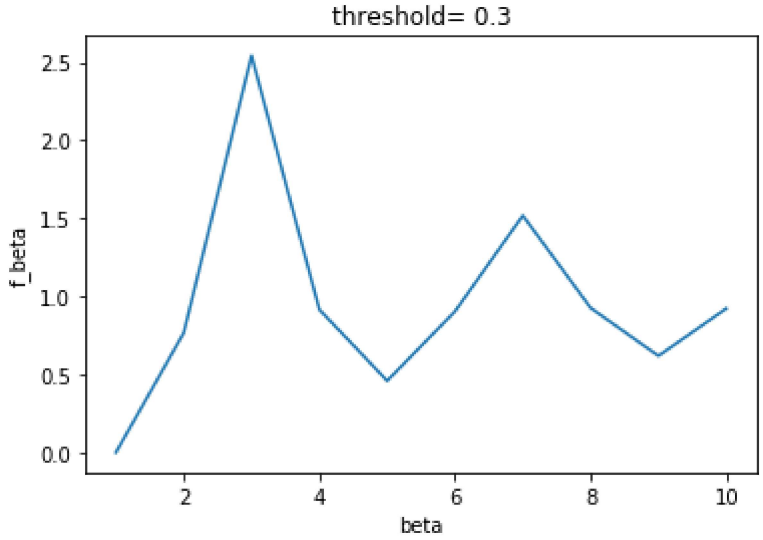


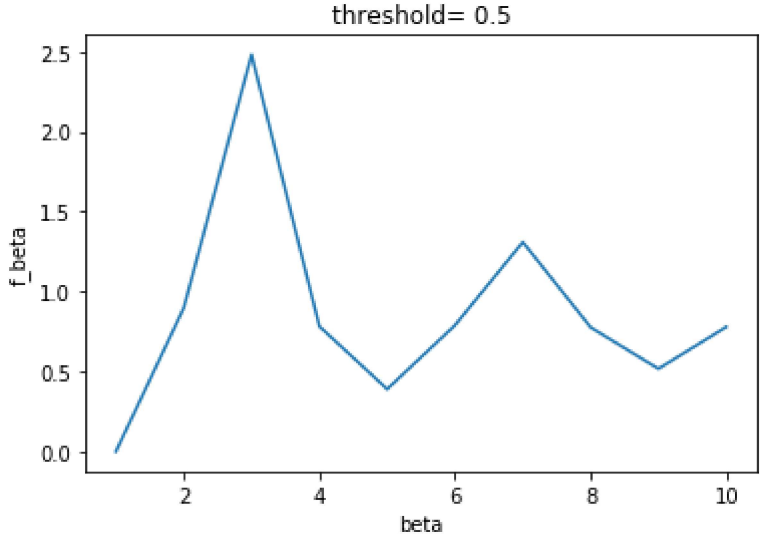
In [45]:

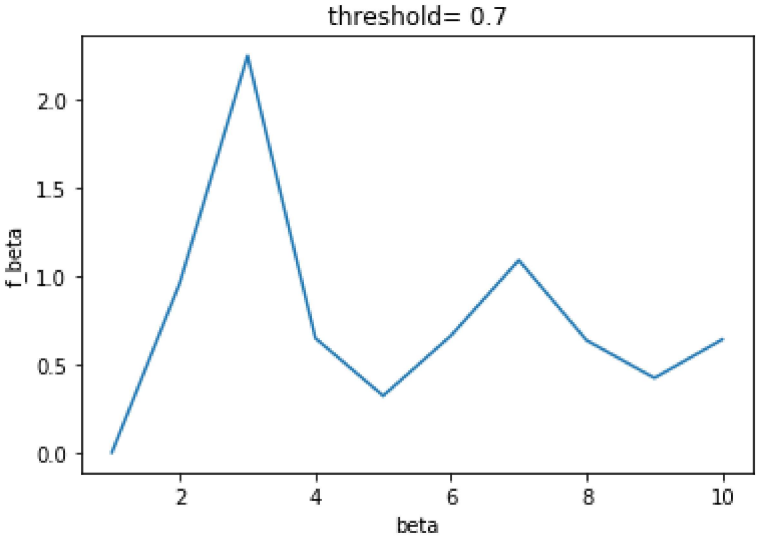
```
def f_beta(precision, recall, beta):
    return ((1+beta)^2)*(precision*recall)/(((beta^2)*precision)+recall)
```

In [46]:

```
threshold_2 = [0.3, 0.5, 0.7]
beta = list(range(1, 11))
res_f_beta = np.zeros(len(beta))
for t in threshold_2:
    for i in range(len(beta)):
        recall = micro_avg_recall(X, y, all_target_class_dict, t)
        precision = micro_avg_precision(X, y, all_target_class_dict, t)
        res_f_beta[i] = f_beta(precision, recall, beta[i])
    plt.figure()
    plt.plot(beta,res_f_beta)
    plt.xlabel('beta')
    plt.ylabel('f_beta')
    plt.title('threshold= {}'.format(t))
```


threshold= 0.3
f_beta
beta
0.0
0.5
1.0
1.5
2.0
2.5
2
4
6
8
10


threshold= 0.5
f_beta
beta
0.0
0.5
1.0
1.5
2.0
2.5
2
4
6
8
10



In [ ]: