Лабораторная работа №2

по курсу "Интеллекутальный анализ данных"

Тема: Логистическая регрессия. Метод опорных векторов. Калибровочные кривые. Отбор признаков.

Выполнила: Синдицкая Виктория

Группа: М80-103М-19

О задании

В этом задании вы:

- настроите метод опорных векторов, визуализируете опорные вектора
- познакомитесь с калибровочными кривыми и сравните вероятности, выдаваемые логистической регрессией и методом опорных векторов
- изучите методы работы с категориальными переменными
- в качестве бонуса попробуете библиотеку vowpal wabbit.

Оценивание и штрафы

Каждая из задач имеет определенную «стоимость» (указана в скобках около задачи). Максимально допустимая оценка за работу — 10 баллов.

Сдавать задание после указанного срока сдачи нельзя. При выставлении неполного балла за задание в связи с наличием ошибок на усмотрение проверяющего предусмотрена возможность исправить работу на указанных в ответном письме условиях.

Задание выполняется самостоятельно. «Похожие» решения считаются плагиатом и все задействованные студенты (в том числе те, у кого списали) не могут получить за него больше 0 баллов (подробнее о плагиате см. на странице курса). Если вы нашли решение какого-то из заданий (или его часть) в открытом источнике, необходимо указать ссылку на этот источник в отдельном блоке в конце вашей работы (скорее всего вы будете не единственным, кто это нашел, поэтому чтобы исключить подозрение в плагиате, необходима ссылка на источник).

Неэффективная реализация кода может негативно отразиться на оценке.

Постановка задачи

- сгенерировать синтетические данные для обучения классификаторов;
- выполнить маштабирование данных;
- обучить классификатор SVM;
- построить ROC и precision-recall кривые; вычислить площадь под кривыми;
- выполнить визуализацию работы алгоритма SVM на двумерных данных; визуализировать опорные векторы;
- обучить модель логистической регрессии;
- построить калибровочные кривые для SVM и логистической регрессии;
- выполнить обучение SVM с калибровкой вероятностей (CalibratedClassifierCV);

In [20]:
```
%pylab inline
import pandas as pd

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
/home/vika/anaconda3/lib/python3.7/site-packages/IPython/core/magics/pylab.py:160: UserWarning: pylab import has cl
obbered these variables: ['ylim', 'clf', 'xlim']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

Часть 1. Метод опорных векторов и калибровка вероятностней

Сгенерируем синтетические данные.

In [21]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

X, y = make_classification(
    n_samples=10000, n_features=20, n_informative=10, n_redundant=10,
    random_state=42)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)

scaler = MinMaxScaler()
X_train_sc = scaler.fit_transform(X_train)
X_test_sc = scaler.transform(X_test)
```

Задание 1. Обучение и применение метода опорных векторов.

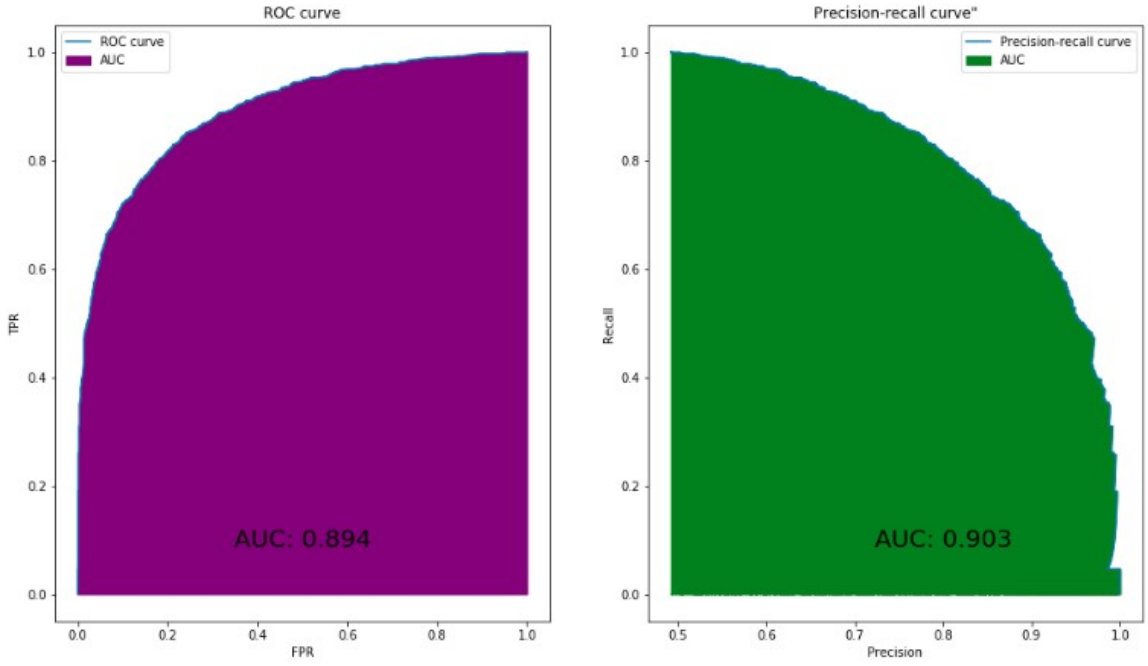
(1 балл)

Обучите метод опорных векторов. На занятиях мы проходили обычный вариант, что соответствует линейному ядру (LinearSVC/LinearSVR в scikit-learn).

In [22]:
```
from sklearn.svm import SVC


clf = SVC(kernel='linear', max_iter=10000, probability=True).fit(X_train_sc, y_train)
```

На тестовой части посчитайте ROC-AUC, PR-AUC. Постройте ROC и PR кривые.

In [24]:
```
from sklearn.metrics import roc_curve, auc, precision_recall_curve
import matplotlib.pyplot as plt

y_pred = clf.predict(X_test_sc)
y_pred_proba = clf.predict_proba(X_test_sc)

fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba[:, 1])
roc_auc = auc(fpr, tpr)
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_proba[:, 1])
pr_auc = auc(recall, precision)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 9))

ax1.plot(fpr, tpr, label='ROC curve')
ax1.fill_between(fpr, tpr, color='purple', label='AUC')
ax1.text(0.5, 0.1, f'AUC: {roc_auc:01.3f}', size=20, ha='center', va='center')
ax1.legend()

ax1.set_title('ROC curve')
ax1.set(xlabel='FPR', ylabel='TPR')

ax2.plot(precision, recall, label='Precision-recall curve')
ax2.fill_between(precision, recall, color='green', label='AUC')
ax2.text(0.8, 0.1, f'AUC: {pr_auc:01.3f}', size=20, ha='center', va='center')
ax2.set_title('Precision-recall curve"')
ax2.set(xlabel='Precision', ylabel='Recall')
ax2.legend()

plt.show()
```



В названии метода присутствуют некоторые "опорные векторы". Сгенерируйте синтетический датасет с помощью make_classification с 2 признаками, обучите на нём метод опорных векторов. Визуализируйте разделяющую прямую, все объекты и выделите опорные вектора (атрибут support_vectors_). В этот раз вместо LinearSVC воспользуйтесь SVC с линейным ядром (kernel='linear'), так как только в нём есть информация об опорных векторах.

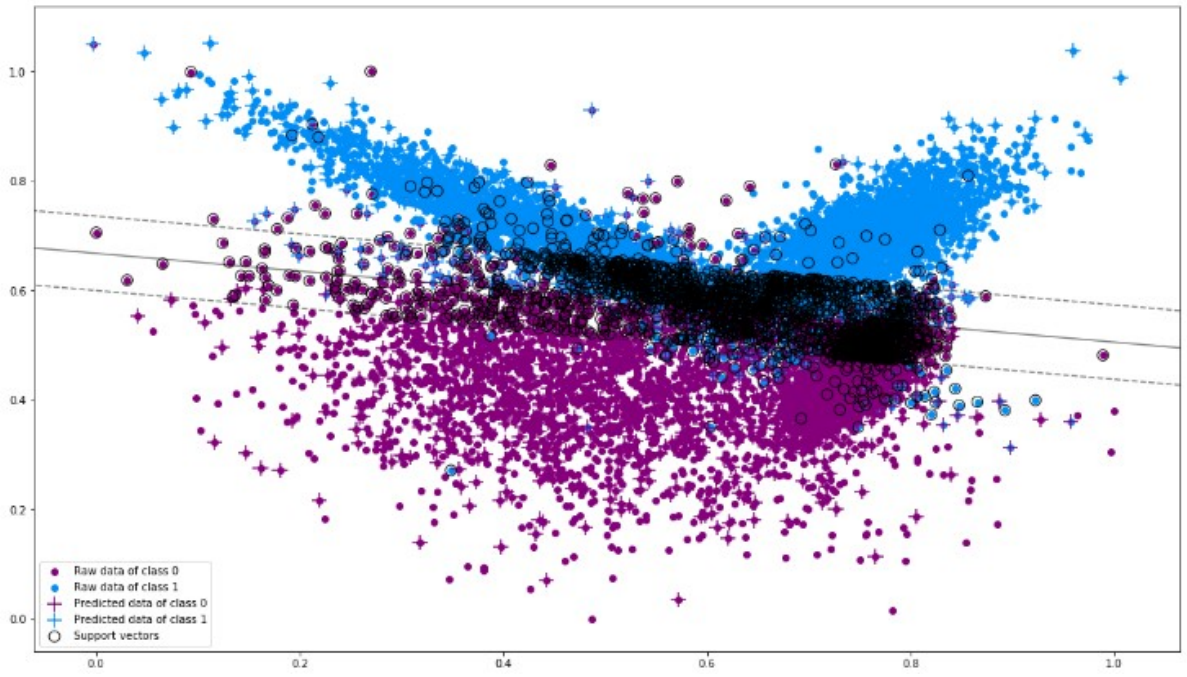
In [26]:
```
X, y = make_classification(
    n_samples=10000, n_features=2, n_informative=2, n_redundant=0,
    random_state=42)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)

scaler = MinMaxScaler()
X_train_sc = scaler.fit_transform(X_train)
X_test_sc = scaler.transform(X_test)
X_sc = scaler.transform(X)
```

In [27]:
```
clf = SVC(kernel='linear', max_iter=10000, probability=True).fit(X_train_sc, y_train)
```

In [28]:
```
y_pred_proba = clf.predict_proba(X_test_sc)
y_pred = clf.predict(X_test_sc)
```

In [29]:

```
plt.figure(figsize=(19,11))

cdict = {0: 'purple', 1: 'dodgerblue'}
for cls in np.unique(y):
    mask1 = y == cls
    plt.scatter(X_sc[mask1, 0], X_sc[mask1, 1], c=cdict[cls], label=f'Raw data of class {cls}')


for cls in np.unique(y):
    mask2 = y_test == cls
    colors = [cdict[c] for c in y_pred[mask2]]
    plt.scatter(X_test_sc[mask2, 0], X_test_sc[mask2, 1], marker='+', s=200, facecolors='none', c=colors,
           label=f'Predicted data of class {cls}')


ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()


xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = clf.decision_function(xy).reshape(XX.shape)


cs = ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])
level_names = {0: 'Hyperplane', 1: 'Class 1', -1: 'Class 0'}
cs.levels = [level_names[val] for val in cs.levels]
ax.clabel(cs, cs.levels, inline=True, fontsize=10)

ax.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1], s=100,
           linewidth=1, facecolors='none', edgecolors='k', label='Support vectors')

plt.legend()
plt.show()
```



Задание 2. Калибровка вероятностей.

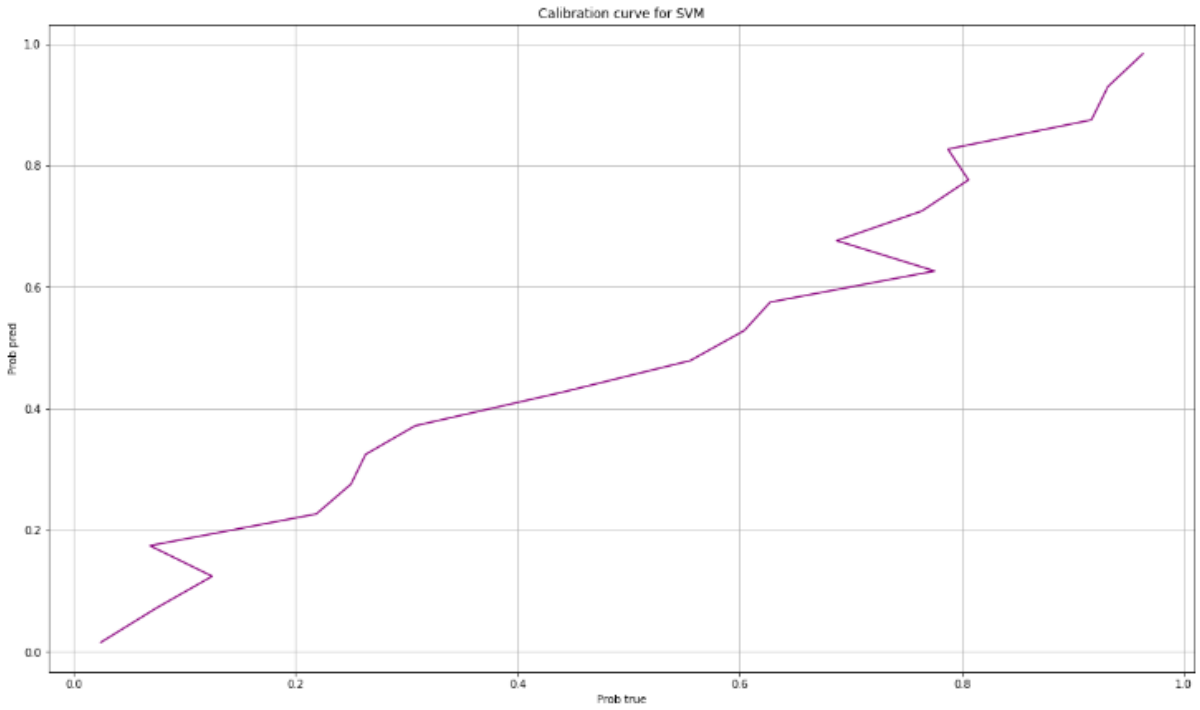
(1.5 балла)

Перейдём к оценке качества выдаваемых алгоритмами вероятностей. Начнём с калибровочных кривых.

Допустим, алгоритм возвращает некоторые числа от нуля до единицы. Хорошо ли они оценивают вероятность? Для этого разобьем отрезок $[0, 1]$ на несколько маленьких отрезков одинаковой длины. Рассмотрим i-й отрезок с границами $[a_i, b_i]$ и предсказания $p_1, p_2, \ldots, p_k$, которые попали в него. Пусть им соответствуют истинные ответы $y_1, y_2, \ldots, y_k$. Если алгоритм выдает корректные вероятности, то среди этих истинных ответов должно быть примерно $(a_i + b_i)/2$ единиц. Иными словами, если нарисовать кривую, у которой по оси X отложены центры отрезков, а по оси Y — доли единичных ответов этих в отрезках, то она должна оказаться диагональной. Ниже приведена функция, которая должна рисовать такие кривые. В ней допущено две ошибки — найдите и исправьте их.

In [31]:
```
from sklearn.calibration import calibration_curve

prob_true, prob_pred = calibration_curve(y_test, y_pred_proba[:, 1], n_bins=20)
```

In [32]:
```
plt.figure(figsize=(19,11))

plt.plot(prob_true, prob_pred, c='purple')
plt.title('Calibration curve for SVM')
plt.xlabel('Prob true')
plt.ylabel('Prob pred')
plt.grid()
```



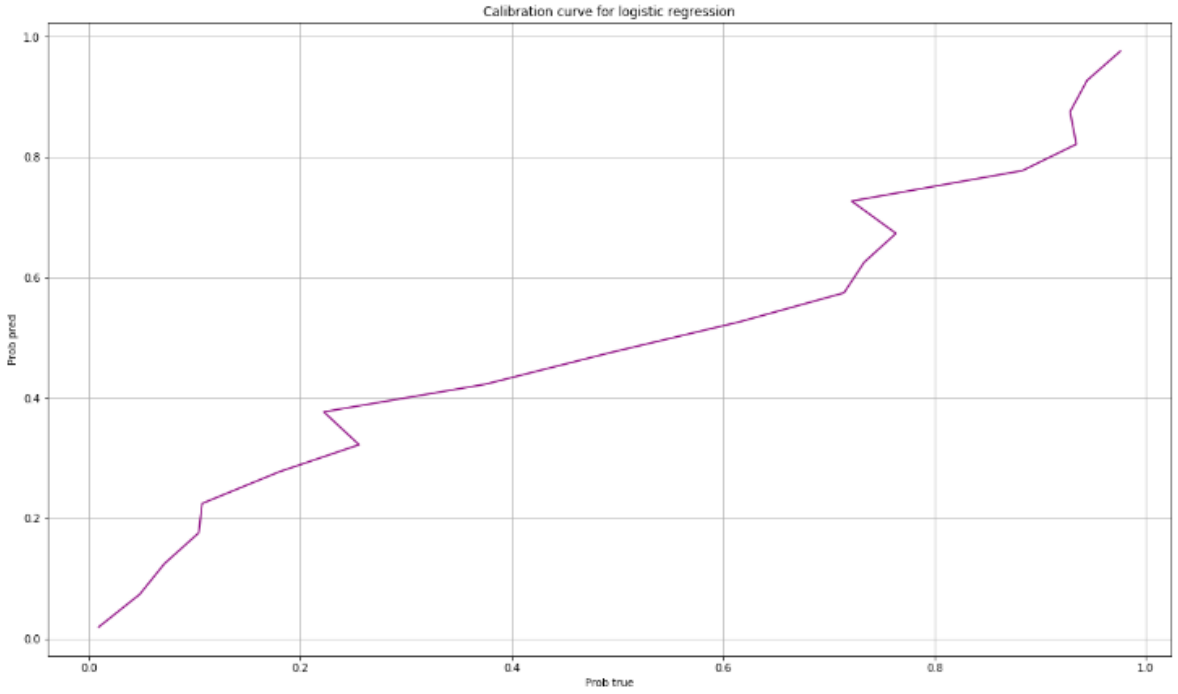
Постройте калибровочные кривые для логистической регрессии и метода опорных векторов. Изучите распределение ответов классификаторов (постройте гистограммы с помощью plt.hist). Чем они различаются? Чем вы можете объяснить это?

Заметим, что метод опорных векторов не умеет predict_proba, но имеет метод decision_function.

In [33]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression().fit(X_train_sc, y_train)
y_pred_logreg = clf.predict(X_test_sc)
y_pred_proba_logreg = clf.predict_proba(X_test_sc)
```

```
/home/vika/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solv
er will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [34]:
```
prob_true, prob_pred = calibration_curve(y_test, y_pred_proba_logreg[:, 1], n_bins=20)

plt.figure(figsize=(19,11))

plt.plot(prob_true, prob_pred, c='purple')
plt.title('Calibration curve for logistic regression')
plt.xlabel('Prob true')
plt.ylabel('Prob pred')
plt.grid()
```



Воспользуйтесь CalibratedClassifierCV из sklearn для калибровки вероятностей метода опорных векторов на обучении и постройте с его помощью предсказания для тестовой выборки. Нарисуйте для них калибровочную кривую. Улучшилась ли она?

In [35]:
```
from sklearn.calibration import CalibratedClassifierCV

svm = SVC(kernel='linear', max_iter=10000, probability=True)
clf = CalibratedClassifierCV(svm).fit(X_train_sc, y_train)
```

```
/home/vika/anaconda3/lib/python3.7/site-packages/sklearn/model_selection/_split.py:1978: FutureWarning: The defaul
t value of cv will change from 3 to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

In [36]:
```
y_pred_proba = clf.predict_proba(X_test_sc)
y_pred = clf.predict(X_test_sc)
```

In [37]:
```
prob_true, prob_pred = calibration_curve(y_test, y_pred_proba[:, 1], n_bins=20)

plt.figure(figsize=(19,11))

plt.plot(prob_true, prob_pred, c='purple')
plt.title('Calibration curve for logistic SVM after CalibratedClassifierCV')
plt.xlabel('Prob true')
plt.ylabel('Prob pred')
plt.grid()
```