3.5 Метод опорных векторов

Метод опорных векторов (SCM - support vector machine)

Метод используется как для классификации, так и для регрессии.

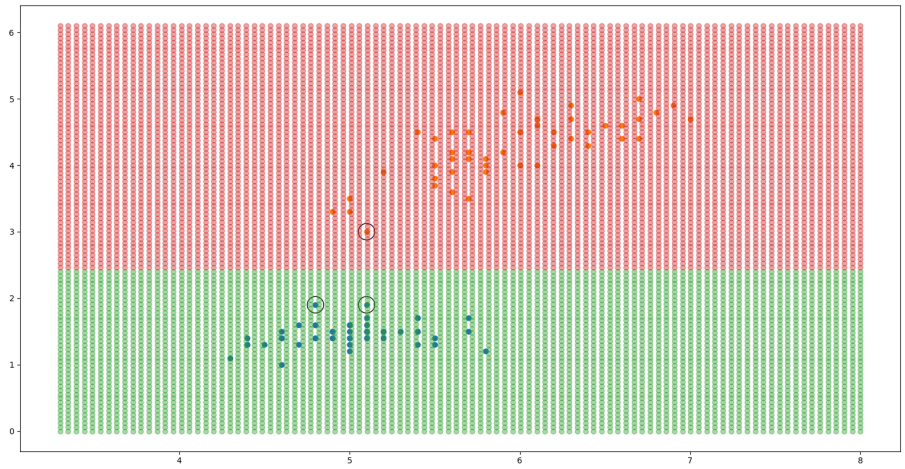
Разделяющая классификация

Рисует некоторую линию, которая разделяет классы данных.

Выбирается линия с макс. отступом.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

iris = sns.load_dataset("iris")
data = iris[["sepal_length", "petal_length", "species"]]
data_df = data[
    (data["species"] == "setosa") | (data["species"] == "versicolor")
]

X = data_df[["sepal_length", "petal_length"]]
y = data_df["species"]

data_df_setosa = data_df[data_df["species"] == "setosa"]
data_df_versicolor = data_df[data_df["species"] == "versicolor"]

plt.scatter(
    data_df_setosa["sepal_length"],
    data_df_setosa["petal_length"],
)
plt.scatter(
    data_df_versicolor["sepal_length"],
    data_df_versicolor["petal_length"],
)
```

```
model = SVC(kernel='linear', C=10_000)
model.fit(X, y)

print(model.support_vectors_)
# [[4.8 1.9]
# [5.1 1.9]
# [5.1 3. ]]

plt.scatter(model.support_vectors_[:,0],
            model.support_vectors_[:,1],
            s=400,
            facecolor='none',
            edgecolors="black")

x1_p = np.linspace(
    data_df["sepal_length"].min() - 1, data_df["sepal_length"].max() + 1, 100
)
x2_p = np.linspace(
    data_df["petal_length"].min() - 1, data_df["petal_length"].max() + 1, 100
)
x1_p, x2_p = np.meshgrid(x1_p, x2_p)
X_p = pd.DataFrame(
    np.c_[x1_p.ravel(), x2_p.ravel()],
    columns=["sepal_length", "petal_length"]
)

y_p = model.predict(X_p)
X_p["species"] = y_p

plt.scatter(
    X_p[X_p["species"] == "setosa"]["sepal_length"],
    X_p[X_p["species"] == "setosa"]["petal_length"],
    alpha=0.4
)
plt.scatter(
    X_p[X_p["species"] == "versicolor"]["sepal_length"],
    X_p[X_p["species"] == "versicolor"]["petal_length"],
    alpha=0.4
)

plt.show()
```



Практическое задание

убрать из данных iris часть точек (на которых мы обучаемся) и убедиться что на предсказание влияют только опорные вектора.

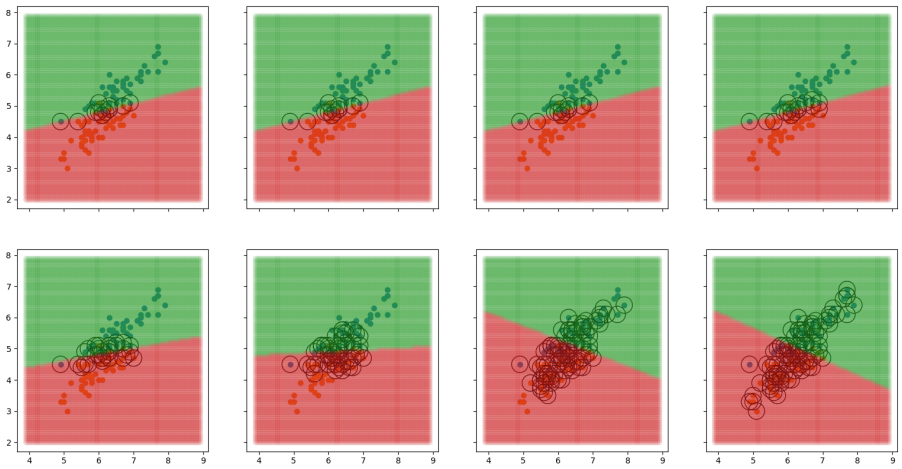
Перекрытие данных

В случае, если данные перекрываются, то понятно что идеальной границы не существует. У модели существует гиперпараметров, который определяет “размытие” отступа

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

iris = sns.load_dataset("iris")
data = iris[["sepal_length", "petal_length", "species"]]
data_df = data[
    (data["species"] == "virginica") | (data["species"] == "versicolor")
]


X = data_df[["sepal_length", "petal_length"]]
y = data_df["species"]

data_df_virginica = data_df[data_df["species"] == "virginica"]
```

```
data_df_versicolor = data_df[data_df["species"] == "versicolor"]

c_value = [[10_000, 1_000, 100, 10], [1, 0.1, 0.01, 0.001]]
fig, ax = plt.subplots(2, 4, sharex='col', sharey='row')

for i in range(2):
   for j in range(4):

      ax[i, j].scatter(
         data_df_virginica["sepal_length"],
         data_df_virginica["petal_length"],
      )
      ax[i, j].scatter(
         data_df_versicolor["sepal_length"],
         data_df_versicolor["petal_length"],
      )

      # Если C большое, то отступ задается "жестко", чем меньше C,
      тем отступ становится более "размытым"
      model = SVC(kernel='linear', C=c_value[i][j])
      model.fit(X, y)


      ax[i, j].scatter(model.support_vectors_[:,0],
               model.support_vectors_[:,1],
               s=400,
               facecolor='none',
               edgecolors="black")

      x1_p = np.linspace(
      data_df["sepal_length"].min() - 1, data_df["sepal_length"].max() + 1, 100
      )
      x2_p = np.linspace(
         data_df["petal_length"].min() - 1, data_df["petal_length"].max() + 1, 100
      )
      x1_p, x2_p = np.meshgrid(x1_p, x2_p)
      X_p = pd.DataFrame(
         np.c_[x1_p.ravel(), x2_p.ravel()],
         columns=["sepal_length", "petal_length"]
      )

      y_p = model.predict(X_p)
      X_p["species"] = y_p

      ax[i, j].scatter(
```

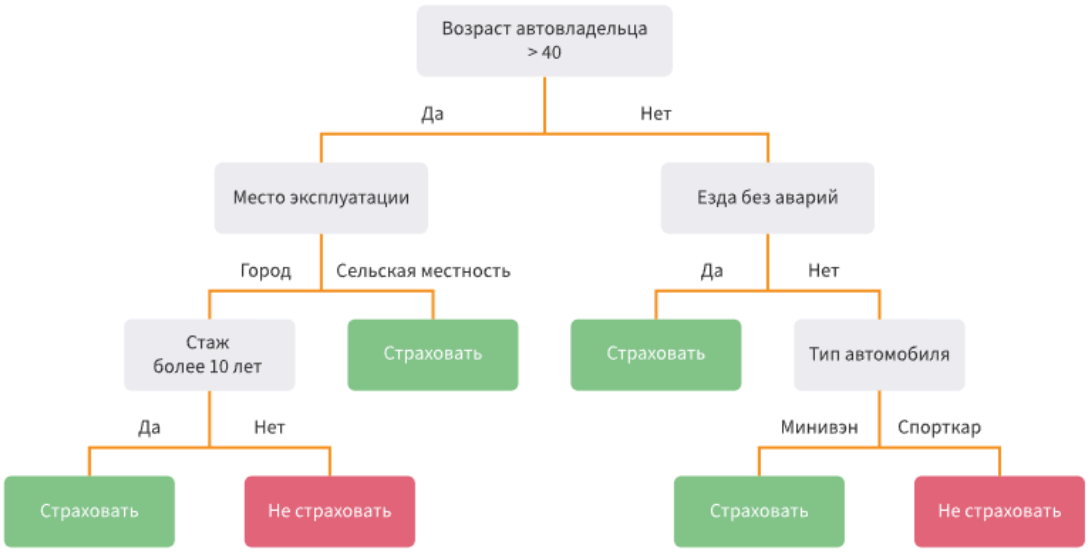
```
            X_p[X_p["species"] == "virginica"]["sepal_length"],
            X_p[X_p["species"] == "virginica"]["petal_length"],
            alpha=0.1
        )
        ax[i, j].scatter(
            X_p[X_p["species"] == "versicolor"]["sepal_length"],
            X_p[X_p["species"] == "versicolor"]["petal_length"],
            alpha=0.1
        )

plt.show()
```



Плюсы	Минусы
Зависимость от небольшого числа опорных векторов ⇒ компактность моделей	При большом количестве обучающих образцов могут быть значительные вычислительные затраты
После обучения предсказания проходит очень быстро	Большая зависимость от размытости C. Поиск может привести к большим вычислительным затратам
На работу метода влияют ТОЛЬКО точки, находящиеся только возле отступов, поэтому эти методы подходят для многомерных данных	У результатов отсутствует вероятностная интерпретация

Деревья решений и случайные леса

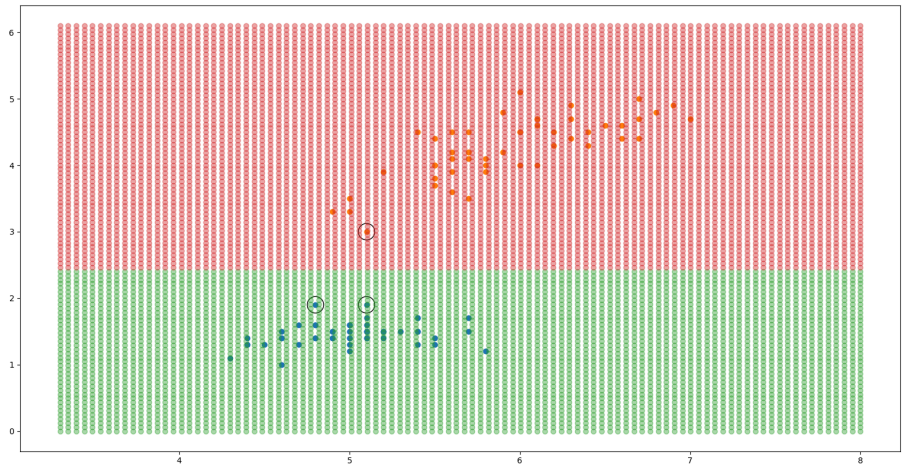
Случайные леса - это непараметрический алгоритм, пример ансамблевого метода, основанного на агрегации результатов множества простых моделей.



В реализациях дерева принятия решений в машинном обучении, вопросы ведут к разделению данных по осям, то есть каждый узел разбивает данные на две группы по одному из признаков.

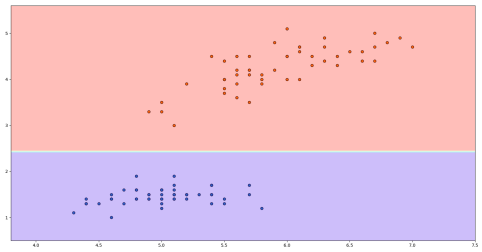
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier

iris = sns.load_dataset("iris")
data = iris[["sepal_length", "petal_length", "species"]]
data_df = data[
    (data["species"] == "setosa") | (data["species"] == "versicolor")
]

X = data_df[["sepal_length", "petal_length"]]
y = data_df["species"]

data_df_setosa = data_df[data_df["species"] == "setosa"]
data_df_versicolor = data_df[data_df["species"] == "versicolor"]
plt.scatter(
    data_df_setosa["sepal_length"],
    data_df_setosa["petal_length"],
    label='Setosa (реальные)',
    edgecolor='black'
)
plt.scatter(
```

```
    data_df_versicolor["sepal_length"],
    data_df_versicolor["petal_length"],
    label='Versicolor (реальные)',
    edgecolor='black'
)

model = DecisionTreeClassifier()
model.fit(X, y)

x1_p = np.linspace(
    data_df["sepal_length"].min() - 1, data_df["sepal_length"].max() + 1, 100
)
x2_p = np.linspace(
    data_df["petal_length"].min() - 1, data_df["petal_length"].max() + 1, 100
)
x1_p, x2_p = np.meshgrid(x1_p, x2_p)
X_p = pd.DataFrame(
    np.c_[x1_p.ravel(), x2_p.ravel()],
    columns=["sepal_length", "petal_length"]
)

y_p = model.predict(X_p)
X_p["species"] = y_p

plt.scatter(
    X_p[X_p["species"] == "setosa"]["sepal_length"],
    X_p[X_p["species"] == "setosa"]["petal_length"],
    alpha=0.4
)
plt.scatter(
    X_p[X_p["species"] == "versicolor"]["sepal_length"],
    X_p[X_p["species"] == "versicolor"]["petal_length"],
    alpha=0.4
)

plt.show()
```

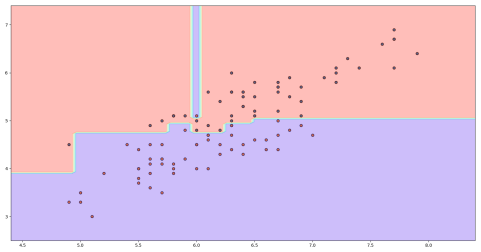



```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier

iris = sns.load_dataset("iris")

# Создаем список с числовыми метками для каждого вида
species_int = []
for r in iris.values:
    match r[4]:
        case "setosa":
            species_int.append(1)
        case "versicolor":
            species_int.append(2)
        case "virginica":
            species_int.append(3)

species_int_df = pd.DataFrame(species_int, columns=["species"])

# Объединяем признаки с числовыми метками
data = iris[["sepal_length", "petal_length"]]
data["species"] = species_int_df

data_df = data[(data["species"] == 1) | (data["species"] == 2)]

X = data_df[["sepal_length", "petal_length"]]
```

```
y = data_df["species"]

model = DecisionTreeClassifier()
model.fit(X, y)


data_df_setosa = data_df[data_df["species"] == 1]
data_df_versicolor = data_df[data_df["species"] == 2]
plt.scatter(
    data_df_setosa["sepal_length"],
    data_df_setosa["petal_length"],
    edgecolor='black',
)
plt.scatter(
    data_df_versicolor["sepal_length"],
    data_df_versicolor["petal_length"],
    edgecolor='black',
)

# Создание сетки для построения разделяющей поверхности
x1_p = np.linspace(X["sepal_length"].min() - 0.5, X["sepal_length"].max() + 0.5, 100)
x2_p = np.linspace(X["petal_length"].min() - 0.5, X["petal_length"].max() + 0.5, 100)
x1_p, x2_p = np.meshgrid(x1_p, x2_p)
X_p = pd.DataFrame(
    np.c_[x1_p.ravel(), x2_p.ravel()],
    columns=["sepal_length", "petal_length"]
)

# Предсказание для каждой точки сетки
y_p = model.predict(X_p)

# Отрисовка разделяющей поверхности с помощью contourf
plt.contourf(
    x1_p, x2_p, y_p.reshape(x1_p.shape),
    alpha=0.3, cmap='rainbow'
)

plt.show()
```



data[(data["species"] == 1) | (data["species"] == 2)]



data[(data["species"] == 3) | (data["species"] == 2)]

max_depth

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier

iris = sns.load_dataset("iris")

species_int = []
for r in iris.values:
    match r[4]:
        case "setosa":
            species_int.append(1)
        case "versicolor":
            species_int.append(2)
        case "virginica":
            species_int.append(3)

species_int_df = pd.DataFrame(species_int, columns=["species"])

data = iris[["sepal_length", "petal_length"]]
data["species"] = species_int_df

data_df = data[(data["species"] == 3) | (data["species"] == 2)]

X = data_df[["sepal_length", "petal_length"]]
y = data_df["species"]

data_df_virginica = data_df[data_df["species"] == 3]
data_df_versicolor = data_df[data_df["species"] == 2]
```

```
max_depth_grid = [[1, 2, 3, 4], [5, 6, 7, 8]]

fig, ax = plt.subplots(2, 4, figsize=(16, 8), sharex='col', sharey='row')

for i in range(2):
    for j in range(4):
        current_depth = max_depth_grid[i][j]

        ax[i, j].scatter(
            data_df_virginica["sepal_length"],
            data_df_virginica["petal_length"],
            edgecolor='black', c='orange',
        )
        ax[i, j].scatter(
            data_df_versicolor["sepal_length"],
            data_df_versicolor["petal_length"],
            edgecolor='black', c='blue',
        )

        # Обучение модели с текущей глубиной
        model = DecisionTreeClassifier(max_depth=current_depth)
        model.fit(X, y)

        # Создание сетки для построения разделяющей поверхности
        x1_p = np.linspace(X["sepal_length"].min() - 0.5,
                           X["sepal_length"].max() + 0.5, 100)
        x2_p = np.linspace(X["petal_length"].min() - 0.5,
                           X["petal_length"].max() + 0.5, 100)
        x1_p, x2_p = np.meshgrid(x1_p, x2_p)
        X_p = pd.DataFrame(
            np.c_[x1_p.ravel(), x2_p.ravel()],
            columns=["sepal_length", "petal_length"]
        )

        y_p = model.predict(X_p)

        ax[i, j].contourf(
            x1_p, x2_p, y_p.reshape(x1_p.shape),
            alpha=0.3, cmap='rainbow'
        )


plt.show()
```