Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Katedra Kybernetiky a Umelej Inteligencie

Stroje s Podpornými Vektormi

Diplomová práca

Inteligentné systémy

Používateľská príručka

Vedúci diplomovej práce:

Ing. Miroslava Matejová

Diplomant:

Roman Dzhulai

Konzultant diplomovej práce:

Ing. Miroslava Matejová

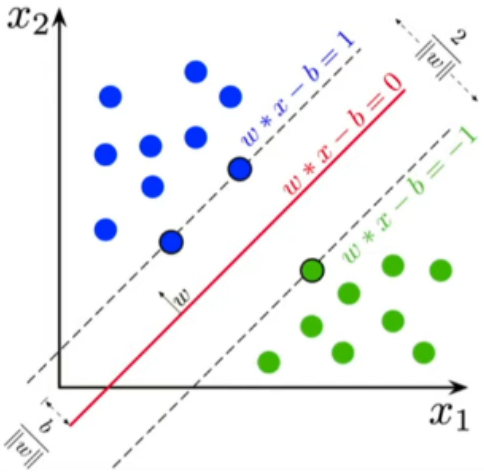
Košice 2024

Obsah

1 Teoretický popis algoritmu

1.1 Nápad

Použitie lineárneho modelu a snaha nájsť lineárnu rozhodovaciu hranicu (hyperrovina), ktorá najlepšie oddelí dáta. Najlepšia hyperrovina je tá, ktorá poskytuje najväčšie oddelenie/margín medzi oboma triedami. Preto vyberieme hyperrovinu tak, aby vzdialenosť od nej k najbližšiemu bodu na oboch stranách bola maximalizovaná.



Obr. 1 – 1 Lineárne SVM

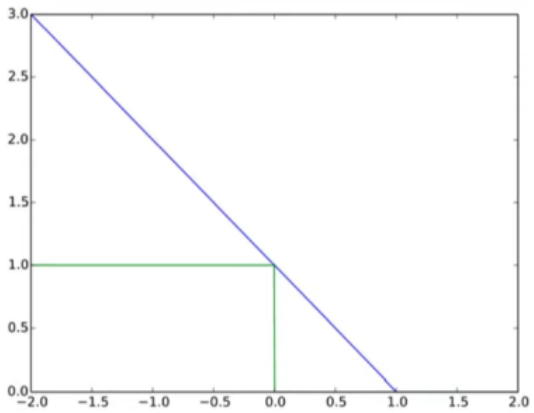
$$w^T x_i - b \geq 1 \text{ ak } y_i = 1$$

$$w^T x_i - b \leq -1 \text{ ak } y_i = -1$$

$$y_i(w^T x_i - b) \geq 1$$

Tu, w je vektor váh, x_i je vstupný vektor, b je bias a y_i je označenie pre bod dát x_i. Tieto rovnice predstavujú rozhodovaciu funkciu pre lineárny SVM. Prvé dve rovnice definujú margíny pre pozitívne a negatívne triedy, resp. Tretia rovnica ich kombinuje do jednej rovnice, ktorá zabezpečuje správnu klasifikáciu všetkých bodov.

1.2 Loss funkcia: Hinge Loss



Obr. 1 – 2 Hinge Loss

$$\ell(y) = \max(0, 1 - y(w^T x_i - b))$$

Kde,

- $\ell(y)$ reprezentuje hinge loss. - $w^T x_i - b$ znamená predikovanú hodnotu. - y_i označuje skutočnú triedu, ktorá je buď 1 alebo -1 v binárnej klasifikácii.

Ak je predpokladané skóre na správnej strane hranice rozhodnutia a je tam rozpätie aspoň 1, strata je nula. Ak je však predpoveď na nesprávnej strane alebo v rámci rozpätia, strata rastie lineárne so vzdialenosťou od hranice rozhodnutia.

1.3 Pridanie regularizácie

Do nášej cost funkcie zahrnieme aj regularizačný člen. Bude naša cost funkcia definovaná ako:

$$J = \lambda ||w||^2 + \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i - b))$$

Kde,

- J reprezentuje cost funkciu. - λ je parameter, ktorý kontroluje dôležitosť regularizačného člena. - $||w||^2$ predstavuje L2 normu váhového vektora w, ktorá je mierou

veľkosti váhového vektora. - n je počet dátových bodov.

Uvedená rovnica zahŕňa kompromis medzi minimalizáciou straty a maximalizáciou vzdialenosti na obe strany hranice rozhodnutia.

1.4 Gradienty

$$\begin{aligned}
&\text{ak } y_i \cdot (w \cdot x) \geq 1 : \\
&\quad \frac{dJ}{dw_k} = 2\lambda w_k \quad \frac{dJ}{db} = 0 \\
&\text{inak:} \\
&\quad \frac{dJ}{dw_k} = 2\lambda w_k - y_i x_i \\
&\quad \frac{dJ}{db} = y_i
\end{aligned}$$

Pri diferenciácii overíme, či sme na správnej strane. Gradienty cost funkcie sú $2\lambda w_k$ ak $y_i \cdot (w \cdot x) \geq 1$, s ohľadom na w_k, a gradienty sú nulové s ohľadom na b, čo naznačuje žiadnu zmenu v tomto prípade. V inom prípade je to náš gradient s ohľadom na váhy $2\lambda w_k - y_i x_i$, zatiaľ čo s ohľadom na b je to y_i.

1.5 Aktualizačné pravidlo

$$\text{ak } y_i \cdot (w \cdot x) \geq 1 :$$

$$w = w - \alpha \cdot \frac{dJ}{dw} = w - \alpha \cdot 2\lambda w$$

$$b = b - \alpha \cdot \frac{dJ}{db} = b$$

$$\text{inak:}$$

$$w = w - \alpha \cdot \frac{dJ}{dw} = w - \alpha \cdot (2\lambda w - y_i \cdot x_i)$$

$$b = b - \alpha \cdot \frac{dJ}{db} = b - \alpha \cdot y_i$$

Počas trénovania so zadanou sadou údajov začíname inicializáciou váh. Uistíme sa, že naše triedne značky sú standardizované na -1 a 1, a postupujeme aplikáciou aktualizačných pravidiel diskutovaných vyššie. Tieto aktualizačné pravidlá zahŕňajú úpravu váh na základe algoritmu gradientovej metódy zostupu, kde sa gradient znižuje faktorom určeným rýchlosťou učenia (označenou ako α). Tento proces sa opakuje po určitý počet iterácií, čo umožňuje modelu naučiť sa a prispôsobiť svoje váhy podľa potreby.

1.6 Kroky

Trénovanie (Naučenie váh):

- Inicializujte váhy: To znamená nastavenie počiatočných hodnôt váh modelu. Tieto váhy sú v podstate parametre, ktoré sa model naučí upravovať, aby vykonával presné predikcie.
- Uistite sa, že $y \in \{-1, 1\}$: Zabezpečuje, že cieľová premenná (y) nadobúda len hodnoty -1 alebo 1.
- Aplikujte aktualizačné pravidlá počas n_iters: Počet iterácií (n_iters) určuje, ako dlho bude prebiehať trénovací proces. Tento proces sa opakuje n_iters-krát, čo znamená, že aktualizačné pravidlá budú aplikované v každej iterácii trénovania.

Predikcia:

- Vypočítajte $y = \text{sign}(w \cdot x - b)$: Tento krok vypočíta predikovanú hodnotu (y) pre nový vstup (x) pomocou naučených váh (w) a posunu (b) modelu. Funkcia sign určuje triednu značku (+1 alebo -1) na základe váženej sumy vstupných príznakov a posunu.

2 Vysvetlenie kódu

2.1 Trieda SVM

Táto trieda definuje model Support Vector Machine (SVM) pre binárnu klasifikáciu.

2.1.1 Inicializácia

```
class SVM:
    def __init__(self, lr=0.0001, C=0, tol=1e-7, max_iter=10000, verbose=False):
        """
        Inicializuje model Support Vector Machine (SVM).

        Parametre:
        - lr: Rýchlosť učenia pre gradientný zostup.
        - C: Regularizačný parameter.
        - tol: Tolerancia na určenie konvergencie.
        - max_iter: Maximálny počet iterácií.
        - verbose: Určuje, či sa má počas trénovania vypisovať pokrok.
        """
        self.lr = lr
        self.C = C
        self.tol = tol
        self.max_iter = max_iter
        self.verbose = verbose
        self.w = None
```

Listing 1 Inicializácia triedy SVM

Funkcia `__init__`, známa tiež ako konštruktor, v tomto kúsku kódu je zodpovedná za inicializáciu inštancie modelu SVM. Tu je rozbor toho, čo sa deje vnútri

funkcie:

Definuje parametre:

- `lr` (miera učenia): Toto ovláda veľkosť kroku počas trénovania pomocou gradientného zostupu (štandardne je 0.0001).
- `C` (paramater regularizácie): Toto určuje kompromis medzi prispôsobením trénovacích dát a zložitosťou modelu (štandardne je 0).
- `tol` (tolerancia): Toto nastavuje prah pre považovanie modelu za zkonvergovaný počas trénovania (štandardne je 1e-7).
- `max_iter` (maximálne iterácie): Toto obmedzuje počet trénovacích iterácií, aby sme sa nezasekli (štandardne je 10000).
- `verbose` (hlučnosť): Toto riadi, či sa tlačia správy o priebehu trénovania (štandardne `False`).

Alokácia atribútov:

- Vytvára inštancie premenných v rámci objektu na uchovávanie týchto hyperparametrov:
  - `self.lr`: Uchováva mieru učenia.
  - `self.C`: Uchováva parameter regularizácie.
  - `self.tol`: Uchováva hodnotu tolerancie.
  - `self.max_iter`: Uchováva maximálny počet iterácií.
  - `self.verbose`: Uchováva nastavenie hlučnosti.
- Okrem toho vytvára atribút s názvom `self.w` a inicializuje ho na `None`. Táto premenná neskôr udrží váhový vektor modelu, ale počas inicializácie ešte nie je k dispozícii.

2.1.2 Rozhodovacia funkcia

```
def decision_function(self, X):
        """
        Vypočíta hodnotu rozhodovacej funkcie pre vstupné dá
   ta.

        Parametre:
        - X: Vstupné dáta.

        Vráti:
        - Hodnoty rozhodovacej funkcie.
        """
        return np.dot(X, self.w[1:]) - self.w[0]
```

Listing 2 Rozhodovacia funkcia triedy SVM

Táto funkcia vypočíta hodnotu rozhodovacej funkcie pre vstupné dáta. Rozhodovacia funkcia je skóre, ktoré model SVM priradí každému vstupnému bodu. Hodnota skóre naznačuje, ako silno model klasifikuje bod do jednej z dvoch tried.

Parametre:

- `X`: Vstupné dáta. Matica s rozmermi (`n_samples`, `n_features`), kde `n_samples` je počet vstupných bodov a `n_features` je počet funkcií.

Návratová hodnota:

- Hodnoty rozhodovacej funkcie: Pole s dĺžkou `n_samples`, kde každý prvok je skóre pre príslušný vstupný bod.

Popis:

- **Výpočet skalárneho súčinu:** Funkcia najprv vypočíta skalárny súčin medzi vstupnými dátami a váhovým vektorom modelu (`self.w[1:]`). Váhový vektor je parameter modelu naučený počas trénovania.

- **Odrátanie biasu:** Následne je od skalárneho súčinu odpočítaný bias modelu (`self.w[0]`).

2.1.3 Nákladová funkcia

```
def cost(self, X, y):
        """
        Vypočíta hodnotu nákladovej funkcie.

        Parametre:
        - X: Vstupné dáta.
        - y: Skutočné značky.

        Vráti:
        - Hodnotu nákladovej funkcie.
        """
        y_pred = self.decision_function(X)
        return self.C * np.linalg.norm(self.w[1:]) ** 2 + np.
  mean(hinge_loss(y, y_pred))
```

Listing 3 Nákladová funkcia triedy SVM

Táto funkcia vypočíta hodnotu nákladovej funkcie pre model SVM. Nákladová funkcia sa skladá z dvoch zložiek:

Regularizačný člen: Tento člen trestá model za šírku medzi okrajmi. Vyššia hodnota `C` vedie k silnejšiemu trestaniu a jednoduchšiemu modelu.

Strata z klasifikácie: Tento člen trestá model za nesprávne klasifikované body. Použitá strata je hranová strata, definovaná nasledovne:

$$\text{hinge\_strata}(y, \hat{y}) = \max(0, 1 - y \cdot \hat{y})$$

Nákladová funkcia sa používa na optimalizáciu modelu SVM. Cieľom je minimalizovať hodnotu nákladovej funkcie, aby sme dosiahli model s dobrou správnosťou.

Parametre:

- `X`: Vstupné dáta.
- `y`: Skutočné hodnoty.

Návratová hodnota:

- Hodnota nákladovej funkcie.

2.1.4 Gradientová funkcia

```
def gradient(self, x_i, y_i):
        """
        Vypočíta gradient stratovej funkcie.

        Parametre:
        - x_i: Bod vstupných dát.
        - y_i: Skutočná značka pre daný bod.

        Vráti:
        - Vektor gradientu.
        """
        if y_i * self.decision_function(x_i) >= 1:
            return np.concatenate(([0], 2 * self.C * self.w
[1:]))
        else:
            return np.concatenate(([y_i], (2 * self.C * self.
w[1:] - y_i * x_i)))
```

Listing 4 Funkcia gradientu pre triedu SVM

Táto funkcia vypočíta gradient stratovej funkcie pre model SVM. Stratová funkcia SVM pozostáva z dvoch komponentov. Gradient je vektor, ktorý naznačuje smer, v ktorom rýchlo narastá stratová funkcia. Gradient sa používa v algoritme

gradientového zostupu na aktualizáciu parametrov modelu s cieľom minimalizovať stratovú funkciu. Výpočet gradientu závisí od toho, či je bod klasifikovaný správne alebo nesprávne.

Parametre:

- `x_i`: Vstupný bod dát.
- `y_i`: Skutočná trieda pre daný bod.

Návratová hodnota:

- Vektor gradientu.

2.1.5 Forward funkcia

```
def forward(self, X):
        """
        Vypočíta predpovedané značky pre vstupné dáta.

        Parametre:
        - X: Vstupné dáta.

        Vráti:
        - Predpovedané značky.
        """
        y_pred = self.decision_function(X)
        return np.sign(y_pred)
```

Listing 5 Forward funkcia triedy SVM

Funkcia `forward` prijíma vstupné dáta a používa model SVM na predikciu príslušných značiek. Najprv vypočíta hodnoty rozhodovacej funkcie a potom ich konvertuje na predikované značky pomocou funkcie `np.sign`.

Parametre:

- `X`: Vstupné dáta.

Návratová hodnota:

- Predikované značky.

2.1.6 Trénovancia funkcia

```
def fit(self, X, y, X_val=None, y_val=None):
        """
        Natrénuje model SVM na trénovacích dátach.

        Parametre:
        - X: Trénovacie dáta.
        - y: Skutočné značky pre trénovacie dáta.
        - X_val: Validácia dát.
        - y_val: Skutočné značky pre validáciu dát.

        Vráti:
        - Podporné vektory.
        """
        n_features = X.shape[1]
        self.w = np.zeros(n_features + 1)

        count = 0
        prev_cost = float('inf')  # Inicializácia s hodnotou
  nekonečna pre porovnanie
        while count < self.max_iter:
            for idx, x_i in enumerate(X):
                dw = self.gradient(x_i, y[idx])
                self.w -= self.lr * dw

```

```
24              cost = self.cost(X, y)
25              if abs(prev_cost - cost) < self.tol:
26                  # Ak je zmena nákladov pod toleranciou,
    zastaví trénovanie
27                  break
28
29              if self.verbose and count % 10 == 0:
30                  y_val_pred = self.forward(X_val)
31                  acc = accuracy(y_val, y_val_pred)
32                  print(f'Iterácia {count + 1}: vahy = {self.w
    }, náklad = {cost:.8f}, Správnosť = {acc:.2%}')
33
34              prev_cost = cost
35              count += 1
36
37          if count == self.max_iter:
38              print(f"Počet iterácií presiahol maximálny počet
    {self.max_iter}")
39          else:
40              print(f'Iterácia {count + 1}: vahy = {self.w}, ná
    klad = {cost:.8f}, Správnosť = {acc:.2%}')
41
42          # Identifikuje podporné vektory
43          support_vector_indices = np.where(np.abs(self.
    decision_function(X)) <= 1 + 1e-2)[0]
44          return X[support_vector_indices]
```

Listing 6 Trénovancia funkcia triedy SVM

Inicializácia:

- Inicializuje váhový vektor `self.w` s nulovými hodnotami.

- Nastavuje počítadlo iterácií `count` na 0.
- Nastavuje predchádzajúcu hodnotu nákladov `prev_cost` na nekonečno na porovnanie.

Trénovací cyklus:

- Cyklus pokračuje, pokiaľ:
  - Počet iterácií `count` je menší ako maximálny počet iterácií `self.max_iter`.
  - Rozdiel medzi predchádzajúcimi a aktuálnymi nákladmi je väčší ako tolerancia `self.tol`.
- V každej iterácii, pre každý trénovací príklad `x_i`:
  - Vypočíta gradient `dw` stratovej funkcie vzhľadom k váhovému vektoru.
  - Aktualizuje váhový vektor pomocou algoritmu gradientového zostupu s mierou učenia `self.lr`.

Tlač informácií:

- Ak je zapnutý režim podrobného výstupu (`self.verbose` je `True`), vypíše každých 10 iterácií:
  - Aktuálny počet iterácií.
  - Hodnoty váh `self.w`.
  - Aktuálnu hodnotu nákladov.
  - Správnosť na validačných dátach (ak sú poskytnuté).

Identifikácia podporných vektorov:

- Podporné vektory sú trénovacie príklady, ktoré ležia najbližšie k hyperrovinke.
- Funkcia ich vráti po ukončení trénovania.

2.2 Metriky

2.2.1 Hinge Loss funkcia

```
def hinge_loss(y, y_pred):
    """
    Výpočet hinge straty.

    Args:
        y (numpy.ndarray): Pole pravdivých hodnôt.
        y_pred (numpy.ndarray): Pole predpovedaných hodnôt.

    Returns:
        numpy.ndarray: Hodnoty strát pre jednotlivé príklady.
    """
    return np.max([np.zeros(y.shape[0]), 1 - y * y_pred])
```

Listing 7 Hinge Loss funkcia

Funkcia `hinge_loss` vypočíta závesnú stratu medzi pravdivými hodnotami (`y`) a predpovedanými hodnotami (`y_pred`). V tomto prípade sa využíva vzorec pre závesnú stratu vo viacrozmernej klasifikácii. Ak je predpoveď správna, teda $y \times y_{\text{pred}} > 1$, závesná strata je nulová. Ak je predpoveď nesprávna, závesná strata je daná rozdielom $1 - y \times y_{\text{pred}}$. Následne sa použije funkcia `np.max` na zabezpečenie nulovej hodnoty strany straty, čo je kladná hodnota.

2.2.2 Funkcia správnosti

```
def accuracy(y, y_pred):
    """
    Vypočíta presnosť klasifikácie.

    Args:
```

```
        y (numpy.ndarray): Pole pravdivých hodnôt.
        y_pred (numpy.ndarray): Pole predpovedaných hodnôt.

    Returns:
        float: Správnosť klasifikácie.
    """
    # Indexy pozitívnych tried
    idx = np.where(y_pred == 1)
    # Počet správne klasifikovaných pozitívnych príkladov
    true_positives = np.sum(y_pred[idx] == y[idx])

    # Indexy negatívnych tried
    idx = np.where(y_pred == -1)
    # Počet nesprávne klasifikovaných negatívnych príkladov
    true_negatives = np.sum(y_pred[idx] == y[idx])

    # Celková správnosť
    return float(true_positives + true_negatives) / len(y)
```

Listing 8 Funkcia správnosti

Funkcia `accuracy` vypočíta správnosť klasifikácie porovnávaním pravdivých hodnôt (`y`) s predpovedanými hodnotami (`y_pred`). Indexy pozitívnych a negatívnych tried sú určené pomocou funkcie `np.where`. Následne sa vypočíta počet správne klasifikovaných pozitívnych príkladov a správne klasifikovaných negatívnych príkladov a spočíta sa celková správnosť.

2.2.3 Recall funkcia

```
def recall(y, y_pred):
    """
    Výpočet recallu.
```

```
4
5       Args:
6           y (numpy.ndarray): Pole pravdivých hodnôt.
7           y_pred (numpy.ndarray): Pole predpovedaných hodnôt.
8
9       Returns:
10          float: recall.
11      """
12      # Počet správne klasifikovaných pozitívnych príkladov
13      true_positives = np.sum((y_pred == 1) & (y == 1))
14      # Počet nesprávne klasifikovaných pozitívnych príkladov
15      false_negatives = np.sum((y_pred == -1) & (y == 1))
16      # Výpočet recallu
17      return true_positives / (true_positives + false_negatives
    )
```

Listing 9 Recall funkcia

Funkcia `recall` vypočíta odvolanie (recall) porovnávaním pravdivých hodnôt (`y`) s predpovedanými hodnotami (`y_pred`). V tomto prípade je odvolanie definované ako pomer správne klasifikovaných pozitívnych príkladov k celkovému počtu pozitívnych príkladov. Funkcia `np.sum` je použitá na spočítanie správne klasifikovaných a nesprávne klasifikovaných pozitívnych príkladov.

2.2.4 Funkcia presnosti

```
1 def precision(y, y_pred):
2       """
3       Výpočet presnosti klasifikácie.
4
5       Args:
6           y (numpy.ndarray): Pole pravdivých hodnôt.
```

```
        y_pred (numpy.ndarray): Pole predpovedaných hodnôt.

    Returns:
        float: Presnosť klasifikácie.
    """
    # Počet správne klasifikovaných pozitívnych príkladov
    true_positives = np.sum((y_pred == 1) & (y == 1))
    # Počet nesprávne klasifikovaných negatívnych príkladov
    false_positives = np.sum((y_pred == 1) & (y == -1))
    # Výpočet presnosti
    return true_positives / (true_positives + false_positives
   )
```

Listing 10 Funkcia presnosti

Funkcia `precision` vypočíta presnosť klasifikácie porovnávaním pravdivých hodnôt (`y`) s predpovedanými hodnotami (`y_pred`). Presnosť je definovaná ako pomer správne klasifikovaných pozitívnych príkladov k celkovému počtu pozitívnych príkladov predpovedaných klasifikátorom.

2.2.5 Funkcia F1

```
def f1_score(y, y_pred):
    """
    Výpočet F1 skóre ako harmonického priemeru presnosti a
   odvolania.

    Args:
        y (numpy.ndarray): Pole pravdivých hodnôt.
        y_pred (numpy.ndarray): Pole predpovedaných hodnôt.

    Returns:
        float: F1 skóre.
```

```
    """
    # Výpočet presnosti a odvolania
    prec = precision(y, y_pred)
    rec = recall(y, y_pred)
    # Výpočet F1 skóre
    return 2 * (prec * rec) / (prec + rec)
```

Listing 11 Funkcia F1

Funkcia `f1_score` vypočíta F1 skóre, ktoré je harmonickým priemerom presnosti a odvolania. F1 skóre poskytuje vyvážený pohľad na výkon klasifikátora. Najprv sa vypočíta presnosť a odvolanie pomocou funkcií `precision` a `recall`, a potom sa vypočíta F1 skóre podľa definície: $2 \times (\texttt{precision} \times \texttt{recall}) / (\texttt{precision} + \texttt{recall})$.

2.3 Vizualizácia

2.3.1 Vizualizácia SVM

```
def visualize_svm(X, y, w, support_vectors=None):
    """
    Vizualizuje rozhodovaciu hranicu SVM a podporné vektory.

    Parametre:
        X (numpy.ndarray): Vstupné dátové body.
        y (numpy.ndarray): Označenia vstupných dátových bodov
    .
        w (numpy.ndarray): Koeficienty hyperroviny.
        support_vectors (numpy.ndarray): Podporné vektory.
    """
    def get_hyperplane_value(x, w, offset):
        return (-w[1] * x + w[0] + offset) / w[2]

    fig = plt.figure()
```

```
15      ax = fig.add_subplot(1, 1, 1)
16
17      # Vykreslenie dátových bodov
18      ax.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired,
    marker='o', edgecolors='k')
19
20      # Vykreslenie rozhodovacej hranice
21      x0_1 = np.amin(X[:, 0])
22      x0_2 = np.amax(X[:, 0])
23
24      x1_1 = get_hyperplane_value(x0_1, w, 0)
25      x1_2 = get_hyperplane_value(x0_2, w, 0)
26
27      x1_1_m = get_hyperplane_value(x0_1, w, -1)
28      x1_2_m = get_hyperplane_value(x0_2, w, -1)
29
30      x1_1_p = get_hyperplane_value(x0_1, w, 1)
31      x1_2_p = get_hyperplane_value(x0_2, w, 1)
32
33      ax.plot([x0_1, x0_2], [x1_1, x1_2], "y--")
34      ax.plot([x0_1, x0_2], [x1_1_m, x1_2_m], "k")
35      ax.plot([x0_1, x0_2], [x1_1_p, x1_2_p], "k")
36
37      # Vykreslenie podporných vektorov
38      if support_vectors is not None:
39          ax.scatter(support_vectors[:, 0], support_vectors[:,
    1], s=150, facecolors='none', edgecolors='r')
40
41      x1_min = np.amin(X[:, 1])
42      x1_max = np.amax(X[:, 1])
```

```
43      ax.set_ylim([x1_min - 3, x1_max + 3])
44
45      ax.set_xlabel('X1')
46      ax.set_ylabel('X2')
47      ax.set_title('Rozhodovacia hranica SVM')
48
49      plt.show()
```

Listing 12 Vizualizácia SVM

Táto funkcia sa používa na vizualizáciu rozhodovacej hranice Support Vector Machine (SVM) a prípadne podporných vektorov.

Parametre:

- `X (numpy.ndarray)`: Vstupné dátové body.
- `y (numpy.ndarray)`: Označenia vstupných dátových bodov.
- `w (numpy.ndarray)`: Koeficienty hyperroviny.
- `support_vectors (numpy.ndarray)`: Voliteľné. Podporné vektory.

Funkcia vytvorí graf, na ktorom vykreslí:

- Dátové body `X` s príslušnými označeniami `y`.
- Rozhodovaciu hranicu SVM.
- Prípadné podporné vektory, ak sú poskytnuté.

Graf je ďalej označený a osi sú popísané.

3 Popis dát

Kontext

Online rezervačné kanály pre hotely dramaticky ovplyvnili možnosti rezervácií a správanie zákazníkov. Významný počet hotelových rezervácií je zrušených kvôli zrušeniam alebo absencii hostí. Typické dôvody zrušení zahŕňajú zmenu plánov, kolízie v plánoch atď. Často je to uľahčené možnosťou urobiť tak bez poplatku alebo radšej za nízku cenu, čo je výhodné pre hostí hotelov, ale je to menej žiaduce a potenciálne znižujúce príjmy pre hotely.

Môžeme predpovedať, či zákazník splní rezerváciu alebo ju zruší?

Datový slovník

- **Booking_ID**: Unikátny identifikátor každej rezervácie
- **no_of_adults**: Počet dospelých
- **no_of_children**: Počet detí
- **no_of_weekend_nights**: Počet víkendových nocí (sobota alebo nedeľa), ktoré hosť strávil alebo si rezervoval v hoteli
- **no_of_week_nights**: Počet pracovných nocí (pondelok až piatok), ktoré hosť strávil alebo si rezervoval v hoteli
- **type_of_meal_plan**: Typ stravovacieho plánu rezervovaného zákazníkom
- **required_car_parking_space**: Potrebuje zákazník parkovacie miesto pre auto? (0 - Nie, 1 - Áno)
- **room_type_reserved**: Typ izby rezervovanej zákazníkom. Hodnoty sú zašifrované (kódované) hotelmi INN.
- **lead_time**: Počet dní medzi dátumom rezervácie a dátumom príchodu
- **arrival_year**: Rok príchodu
- **arrival_month**: Mesiac príchodu
- **arrival_date**: Dátum v mesiaci

- **market_segment_type**: Označenie segmentu trhu
- **repeated_guest**: Je zákazník opakujúci sa? (0 - Nie, 1 - Áno)
- **no_of_previous_cancellations**: Počet predchádzajúcich rezervácií, ktoré zákazník zrušil pred aktuálnou rezerváciou
- **no_of_previous_bookings_not_canceled**: Počet predchádzajúcich rezervácií, ktoré zákazník nezrušil pred aktuálnou rezerváciou
- **avg_price_per_room**: Priemerná cena za izbu a deň rezervácie; ceny izieb sú dynamické (v eurách)
- **no_of_special_requests**: Celkový počet špeciálnych požiadaviek zákazníka (napr. vysoké poschodie, výhľad z izby atď.)
- **booking_status**: Príznak indikujúci, či bola rezervácia zrušená alebo nie.

Dataset: Tento dataset nájdete na stránke Kaggle.com[1].

4 Vyhodnotenie

V tejto časti hodnotíme výkon nášho prediktívneho modelu pre zrušenia rezervácií hotelov. Používame niekoľko metrík na posúdenie účinnosti modelu pri predikcii, či zákazník splní svoju rezerváciu alebo ju zruší.

4.1 Hodnotiace metriky

Využívame nasledujúce hodnotiace metriky:

- **Správnosť (Accuracy)**: Podiel správne predikovaných výsledkov (ako splnené rezervácie aj zrušenia) na celkovom počte predikcií.

[1] https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset

- **Presnosť (Precision)**: Presnosť pozitívnych predikcií, teda pomer správne predikovaných zrušení k celkovému počtu predikovaných zrušení.
- **Úplnosť (Recall)**: Schopnosť modelu zachytiť všetky zrušenia, teda pomer správne predikovaných zrušení k všetkým skutočným zrušeniam.
- **F1 Skóre**: Harmonický priemer presnosti a úplnosti, poskytujúci vyvážené zhodnotenie modelu.

4.2 Získané Hodnoty

- **Správnosť (Accuracy)**: 80.10%
- **Presnosť (Precision)**: 81.82%
- **Recall (Úplnosť)**: 90.04%
- **F1 Skóre**: 85.73%

4.3 Interpretácia Výsledkov

Model preukazuje silný výkon naprieč všetkými hodnotiacimi metrikami. Správnosť 80.10% naznačuje, že model správne predikuje významnú časť výsledkov, zatiaľ čo presnosť 81.82% naznačuje, že správne identifikuje zrušenia medzi predikovanými zrušeniami. Okrem toho vysoká úplnosť 90.04% naznačuje, že model efektívne zachytáva väčšinu skutočných zrušení. Vyvážené F1 skóre 85.73% podčiarkuje schopnosť modelu dosiahnuť vyvážený pomer presnosti a úplnosti.

4.4 Závery

Na základe výsledkov hodnotenia dospievame k nasledujúcim záverom:

- Náš prediktívny model preukazuje sľubný výkon pri identifikácii zrušení rezervácií hotelov, čo potvrdzujú vysoká presnosť, úplnosť a F1 skóre.

- Presnosť a úplnosť modelu umožňujú manažmentu hotelov efektívne alokovať zdroje a implementovať proaktívne stratégie na minimalizáciu potenciálnej straty príjmov v dôsledku zrušení.
- Ďalšie doladenie a vylepšenie modelu by mohlo potenciálne zlepšiť jeho prediktívny výkon a prispieť k efektívnejším stratégiám riadenia zrušení v hotelovom priemysle.

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitej literatúry

[1] CERVANTES, J., GARCIA-LAMONT, F., RODRÍGUEZ-MAZAHUA, L., LOPEZ, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, vol. 408, pp. 189–215, 2020. ISSN 0925-2312. DOI: 10.1016/j.neucom.2019.10.118. Available at: https://www.sciencedirect.com/science/article/pii/S0925231220307153

[2] BERWICK, R. An Idiot's guide to Support vector machines (SVMs). Available at: https://web.mit.edu/6.034/wwwbob/svm.pdf