# Lista 4 de IA - Gabriel Azevedo Fernandes

sexta-feira, 12 de abril de 2024 16:30

QUESTÃO 1

BASE DE DADOS

| Dia | Aparência | temperatura | Umidade | Ventando | Jogar |
|---|---|---|---|---|---|
| d1 | Sol | Quente | Alta | Não | Não |
| d2 | Sol | Quente | Alta | Sim | Não |
| d3 | Nublado | Quente | Alta | Não | Sim |
| d4 | Chuva | Agradável | Alta | Não | Sim |
| d5 | Chuva | Fria | Normal | Não | Sim |
| d6 | Chuva | Fria | Normal | Sim | Não |
| d7 | Nublado | Fria | Normal | Sim | Sim |
| d8 | Sol | Agradável | Alta | Não | Não |
| d9 | Sol | Fria | Normal | Não | Sim |
| d10 | Chuva | Agradável | Normal | Não | Sim |
| d11 | Sol | Agradável | Normal | Sim | Sim |
| d12 | Nublado | Agradável | Alta | Sim | Sim |
| d13 | Nublado | Quente | Normal | Não | Sim |
| d14 | Chuva | Agradável | Alta | Sim | Não |

- Aparência = Chuva
- Temperatura = Fria
- Umidade = Normal
- Ventando = Sim

Probabilidades prévias de Jogar:

- P(Jogar=Sim) e P(Jogar=Não)
- P(Aparência=Chuva | Jogar)
- P(Temperatura=Fria | Jogar)
- P(Umidade=Normal | Jogar)
- P(Ventando=Sim | Jogar)

Probabilidades Prévias:

Total de dias: 14
Dias Jogar=Sim: 9
Dias Jogar=Não: 5

P(Jogar=Sim) = 9/14
P(Jogar=Não) = 5/14

| | | |
|---|---|---|
| Para P(Jogar=Sim): | ---------- | Para P(Jogar=Não): |
| P(Aparência=Chuva \| Sim) = 2/9 | ---------- | P(Aparência=Chuva \| Não) = 3/5 |
| P(Temperatura=Fria \| Sim) = 3/9 | ---------- | P(Temperatura=Fria \| Não) = 1/5 |
| P(Umidade=Normal \| Sim) = 6/9 | ---------- | P(Umidade=Normal \| Não) = 1/5 |
| P(Ventando=Sim \| Sim) = 3/9 | ---------- | P(Ventando=Sim \| Não) = 3/5 |

Para P(Jogar=Sim) com as condições dadas:
P(Dados|Sim) x P(Sim) = (2/9) x (3/9) x (6/9) x (3/9) x (9/14)
P(Dados|Sim) x P(Sim) = 0.0053 (aproximadamente)

Para P(Jogar=Não) com as condições dadas:
P(Dados|Não) x P(Não) = (3/5) x (1/5) x (1/5) x (3/5) x (5/14)
P(Dados|Não) x P(Não) = 0.0046 (aproximadamente)

Normalização Probabilidades:

$$P(\text{Sim}|\text{Dados}) = \frac{P(Dados|Sim) x P(Sim)}{P(Dados|Sim) x P(Sim) + P(Dados|Não) x P(Não)}$$

$$P(\text{Sim}|\text{Dados}) = \frac{0.0053}{0.0053 + 0.0046}$$

P(Sim|Dados) = 0.05354 (Aproximadamente)

$$P(\text{Não}|\text{Dados}) = \frac{P(\text{Dados}|\text{Não}) \times P(\text{Não})}{P(\text{Dados}|\text{Sim}) \times P(\text{Sim}) + P(\text{Dados}|\text{Não}) \times P(\text{Não})}$$

$$P(\text{Não}|\text{Dados}) = \frac{0.0046}{0.0053 + 0.0046}$$

P(Não|Dados) = 0.04646 (Aproximadamente)

RESPOSTA: a probabilidade de Jogar é de aproximadamente 53.54% e de Não Jogar é de aproximadamente 46.46%

QUESTÃO 2

```python
import numpy as np
from sklearn.naive_bayes import GaussianNB
# Mapeamento manual dos dados categóricos para numéricos
# Aparência: Sol = 1, Chuva = 0, Nublado = 2
# Temperatura: Quente = 1, Fria = 0, Agradável = 2
# Umidade: Alta = 0, Normal = 1
# Ventando: Não = 0, Sim = 1
# Jogar: Não = 0, Sim = 1
# Base de dados codificada
data = np.array([
    [1, 1, 0, 0, 0],  # Sol, Quente, Alta, Não, Não
    [1, 1, 0, 1, 0],  # Sol, Quente, Alta, Sim, Não
    [2, 1, 0, 0, 1],  # Nublado, Quente, Alta, Não, Sim
    [0, 2, 0, 0, 1],  # Chuva, Agradável, Alta, Não, Sim
    # Adicione todos os outros registros da mesma forma
])
# Separação das características e a variável alvo
X = data[:, :-1]
y = data[:, -1]
# Criação e treinamento do modelo Naive Bayes
model = GaussianNB()
model.fit(X, y)
# Registro específico para a previsão com a codificação aplicada
# Aparência = Chuva, Temperatura = Fria, Umidade = Normal, Ventando = Sim
new_record = np.array([[0, 0, 1, 1]])
# Predição e probabilidade
prediction = model.predict(new_record)
probability = model.predict_proba(new_record)
# Mostrando a previsão e as probabilidades
print(f"Previsão de Jogar: {'Sim' if prediction[0] == 1 else 'Não'}")
print(f"Probabilidade de Jogar='Sim': {probability[0][1]}")
print(f"Probabilidade de Jogar='Não': {probability[0][0]}")
```

QUESTÃO 3

```python
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from scipy.stats import randint as sp_randint
# Matriz de dados completa baseada na imagem fornecida
data = np.array([
    [2, 1, 0, 0, 0],  # d1: Sol, Quente, Alta, Não, Não
    [2, 1, 0, 1, 0],  # d2: Sol, Quente, Alta, Sim, Não
    [1, 1, 0, 0, 1],  # d3: Nublado, Quente, Alta, Não, Sim
```

```
    [0, 2, 0, 0, 1],  # d4: Chuva, Agradável, Alta, Não, Sim
    [0, 0, 1, 0, 1],  # d5: Chuva, Fria, Normal, Não, Sim
    [0, 0, 1, 1, 0],  # d6: Chuva, Fria, Normal, Sim, Não
    [1, 0, 1, 1, 1],  # d7: Nublado, Fria, Normal, Sim, Sim
    [2, 2, 0, 0, 0],  # d8: Sol, Agradável, Alta, Não, Não
    [2, 0, 1, 0, 1],  # d9: Sol, Fria, Normal, Não, Sim
    [0, 2, 1, 0, 1],  # d10: Chuva, Agradável, Normal, Não, Sim
    [1, 2, 1, 1, 1],  # d11: Nublado, Agradável, Normal, Sim, Sim
    [1, 2, 0, 1, 1],  # d12: Nublado, Agradável, Alta, Sim, Sim
    [1, 1, 1, 0, 0],  # d13: Nublado, Quente, Normal, Não, Sim
    [0, 2, 0, 1, 0],  # d14: Chuva, Agradável, Alta, Sim, Não
])
# Separação das características e da variável alvo
X = data[:, :-1]
y = data[:, -1]
# Divisão dos dados em treino e teste para validar o modelo posteriormente
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Definindo o modelo Random Forest
rf = RandomForestClassifier(random_state=42)
# Definindo o espaço de hiperparâmetros para o RandomSearch
param_dist = {
    "max_depth": [3, None],
    "max_features": sp_randint(1, 4),
    "min_samples_split": sp_randint(2, 11),
    "min_samples_leaf": sp_randint(1, 11),
    "bootstrap": [True, False],
    "criterion": ["gini", "entropy"]
}
# Executando o Randomized Search com 3 divisões de validação cruzada
random_search = RandomizedSearchCV(rf, param_distributions=param_dist, n_iter=20, cv=3, random_state=
42)
# Treinando o modelo com o RandomSearch
random_search.fit(X_train, y_train)
# Melhor modelo encontrado
best_rf = random_search.best_estimator_
# Definindo o novo registro para previsão (Aparência=Chuva, Temperatura=Fria, Umidade=Normal,
Ventando=Sim)
new_record = np.array([[0, 0, 1, 1]])
# Predição com o melhor modelo
best_rf_prediction = best_rf.predict(new_record)
best_rf_probability = best_rf.predict_proba(new_record)
# Exibição dos resultados
print(f"Melhor modelo Random Forest: {best_rf}")
print(f"Previsão de Jogar com Random Forest: {'Sim' if best_rf_prediction[0] == 1 else 'Não'}")
print(f"Probabilidade de Jogar='Sim' com Random Forest: {best_rf_probability[0][1]}")
print(f"Probabilidade de Jogar='Não' com Random Forest: {best_rf_probability[0][0]}")
```

QUESTÃO 4

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
# Dados hipotéticos baseados na imagem fornecida.
# Substitua isso pela leitura do arquivo ou pela entrada manual dos dados
data = {
    'Leite': ['Não', 'Sim', 'Não', 'Sim', 'Não', 'Não', 'Não', 'Não', 'Não', 'Não'],
    'Café': ['Sim', 'Não', 'Sim', 'Sim', 'Não', 'Não', 'Não', 'Não', 'Não', 'Não'],
    'Cerveja': ['Não', 'Sim', 'Não', 'Não', 'Sim', 'Não', 'Não', 'Não', 'Não', 'Não'],
    'Pão': ['Sim', 'Sim', 'Sim', 'Sim', 'Não', 'Não', 'Sim', 'Não', 'Não', 'Não'],
    'Manteiga': ['Sim', 'Sim', 'Sim', 'Sim', 'Não', 'Não', 'Não', 'Não', 'Não', 'Sim'],
    'Arroz': ['Não', 'Não', 'Não', 'Não', 'Não', 'Não', 'Não', 'Sim', 'Sim', 'Não'],
    'Feijão': ['Não', 'Não', 'Não', 'Não', 'Não', 'Não', 'Não', 'Não', 'Sim', 'Não']
}
# Criando o DataFrame
df = pd.DataFrame(data)
# Convertendo os dados para formato binário (1 para 'Sim', 0 para 'Não')
df_encoded = df.applymap(lambda x: 1 if x == 'Sim' else 0)
# Encontrando itemsets frequentes com suporte mínimo de 0.3
frequent_itemsets = apriori(df_encoded, min_support=0.3, use_colnames=True)
# Gerando as regras com confiança mínima de 0.8
```

```
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
# Exibindo os itemsets frequentes e as regras de associação
print("Itemsets Frequentes:")
print(frequent_itemsets)
print("\nRegras de Associação:")
print(rules)
```