

Wasserzeichen zur PDF-Datei hinzufügen

Ziel:

Der Nutzer hat die Möglichkeit, ein Wasserzeichen zu einer PDF-Datei hinzuzufügen und dabei aus den folgenden Optionen zu wählen:

1. Wasserzeichen auf allen Seiten der PDF-Datei einfügen.
2. Wasserzeichen auf allen Seiten außer der ersten Seite einfügen.
3. Wasserzeichen auf allen Seiten außer einer bestimmten Seite einfügen.
4. Wasserzeichen auf allen Seiten außer in einem bestimmten Seitenbereich einfügen.
5. Wasserzeichen auf allen Seiten außer den geraden Seiten einfügen.
6. Wasserzeichen auf allen Seiten außer den ungeraden Seiten einfügen.

Python-Libraries:

- Streamlit
- PdfReader
- PdfWriter
- Os

Im folgenden Beispiel wird ein Wasserzeichen auf allen Seiten der hochgeladenen PDF-Datei eingefügt, außer auf der 1., 3. und 5. Seite. Das Ergebnis wird lokal gespeichert, wobei der Nutzer den Speicherort selbst bestimmt.

Wasserzeichen zur PDF-Datei hinzufügen

Eine PDF-Datei auswählen



Drag and drop file here

Limit 200MB per file • PDF

Browse files



Machine Learning.pdf 0.9MB



Ein Wasserzeichen auswählen



Drag and drop file here

Limit 200MB per file • PDF

Browse files



Watermark.pdf 10.2KB



Einen Speicherort auswählen (z.B. D:\Ordner_1\Ordner_2\Ordner_n\)

C:\Users\kamal\OneDrive\Desktop\neu\

☒ Einen neuen Namen vergeben?

Neuer PDF-Name (z.B. PDF mit Wasserzeichen.pdf)

Machine Learning Zusammenfassung.pdf

☒ Gibt es einen Ausschluss?

Wo ist der Ausschluss? (PDF-Datei hat 185 Seiten)

z.B. 2bis5 (von Seite 2 bis Seite 5)

z.B. 2und5 (Seite 2 und Seite 5)

☐ Erste Seite

☐ Seite x

☒ Bereich


☐ Gerade Seiten

☐ Ungerade Seiten

1und3und5

Wasserzeichen hinzufügen

Kamal Badawi

A decorative graphic consisting of two vertical bars, one black and one teal, positioned to the left of the subtitle text.

Zusammenfassung zum Thema
Machine Learning

ML-Projekt Lebenszyklus

1. Problemdefinition und Zielsetzung

Ziel: Klare Definition des Geschäftsproblems und der zu erreichenden Ziele.

Details:

- **Problemidentifikation:** Stellen Sie sicher, dass das Problem präzise und verständlich definiert ist.
- **Zielsetzung:** Bestimmen Sie klare und messbare Ziele. Zum Beispiel könnte das Ziel sein, die Spezies einer Iris-Blume basierend auf ihren Merkmalen vorherzusagen.

Vorteile:

- Klarheit über das Projektziel.
- Bessere Kommunikation mit Stakeholdern.
- Richtige Ressourcenallokation.

Beispiel:

Ziel: Klassifizierung der Iris-Spezies basierend auf den Merkmalen.
Erfolgsmetriken: Genauigkeit, Präzision, Recall

2. Datensammlung

Ziel: Sammeln relevanter Daten aus verschiedenen Quellen.

Details:

- **Datenquellen identifizieren:** Interne Datenbanken, öffentliche Datensätze, Web-Scraping, APIs.
- **Datenextraktion:** Sammeln und Speichern der Daten in einem einheitlichen Format.

Vorteile:

- Zugang zu umfangreichen und vielfältigen Datenquellen.
- Erhöhte Datenqualität und Repräsentativität.

Beispiel:

```
from sklearn.datasets import load_iris

# Laden des Iris-Datensatzes
data = load_iris()
X = data.data # Merkmale
y = data.target # Zielvariable (Spezies)
```

3. Datenaufbereitung und Bereinigung

Ziel: Daten bereinigen und für die Modellierung vorbereiten.

Details:

- **Datenbereinigung:** Umgang mit fehlenden Werten, Entfernen von Ausreißern.
- **Transformation und Normalisierung:** Skalierung der Daten, Codierung kategorialer Variablen.

Vorteile:

- Verbesserte Datenqualität.
- Gleichmäßige Datenverteilung für bessere Modellperformance.

Beispiel:

```
import pandas as pd

# Konvertieren in ein DataFrame für eine einfache Analyse
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Überprüfen auf fehlende Werte
print(df.isnull().sum()) # Keine fehlenden Werte im Iris-
Datensatz
```

4. Explorative Datenanalyse (EDA)

Ziel: Verstehen der Daten und Erkennen von Mustern.

Details:

- **Datenvisualisierung:** Histogramme, Boxplots, Streudiagramme.
- **Statistische Analyse:** Berechnung von Mittelwert, Median, Standardabweichung.

Vorteile:

- Tieferes Verständnis der Daten.
- Erkennung von Mustern und Anomalien.

Beispiel:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Paarweise Plot für die Merkmale
sns.pairplot(df, hue='target', markers=["o", "s", "D"])
plt.show()

# Korrelation der Merkmale
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```

5. Feature Engineering

Ziel: Erstellen und transformieren von Merkmalen zur Verbesserung der Modellleistung.

Details:

- **Merkmalerstellung:** Erstellen neuer Merkmale basierend auf vorhandenen Daten.
- **Feature-Transformation:** Normalisierung, Skalierung, One-Hot-Encoding.

Vorteile:

- Verbesserte Modellleistung.
- Reduzierte Dimensionalität bei Beibehaltung relevanter Informationen.

Beispiel:

```
from sklearn.preprocessing import StandardScaler

# Skalierung der Merkmale
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

6. Modellwahl und Training

Ziel: Wählen geeigneter Algorithmen und Training der Modelle.

Details:

- **Modellauswahl:** Auswahl geeigneter Algorithmen basierend auf dem Problem.

- **Modelltraining:** Trainieren der Modelle mit Trainingsdaten.

Vorteile:

- Bessere Anpassung des Modells an die Daten.
- Vergleich verschiedener Modelle zur Auswahl des besten.

Beispiel:

```
from sklearn.model_selection import train_test_split

# Datenaufteilung in Trainings- und Testdatensätze
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier

# Modellinstanzen erstellen
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Decision Tree': DecisionTreeClassifier(),
    'SVM': SVC(),
    'KNN': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Gradient Boosting': GradientBoostingClassifier()
}

# Modelle trainieren
for name, model in models.items():
    model.fit(X_train, y_train)
```

7. Modellbewertung und -validierung

Ziel: Evaluieren der Modelle und Auswahl des besten Modells.

Details:

- **Modellbewertung:** Verwendung geeigneter Metriken zur Bewertung der Modellleistung.
- **Kreuzvalidierung:** Sicherstellen der Robustheit des Modells.

Vorteile:

- Objektive Bewertung der Modellleistung.
- Auswahl des besten Modells basierend auf den Ergebnissen.

Beispiel:

```
from sklearn.metrics import accuracy_score, classification_report

# Modelle bewerten
for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f'Modell: {name}')
    print(f'Genauigkeit: {accuracy_score(y_test, y_pred):.2f}')
    print(classification_report(y_test, y_pred))
    print("="*60)
```

8. Modelloptimierung und Feinabstimmung

Ziel: Verbesserung der Modellleistung durch Hyperparameter-Optimierung.

Details:

- **Hyperparameter-Tuning:** Verwenden von Grid Search oder Random Search zur Optimierung der Hyperparameter.
- **Feinabstimmung:** Anpassung der Modellparameter zur Verbesserung der Leistung.

Vorteile:

- Verbesserte Modellgenauigkeit und -leistung.
- Vermeidung von Overfitting durch richtige Parametereinstellung.

Beispiel:

```
from sklearn.model_selection import GridSearchCV

# Grid Search für SVM
param_grid_svm = {'C': [0.1, 1, 10], 'gamma': [0.01, 0.1, 1]}
grid_search_svm = GridSearchCV(SVC(), param_grid_svm, cv=5)
grid_search_svm.fit(X_train, y_train)

print(f'Beste Parameter für SVM: {grid_search_svm.best_params_}')
best_svm = grid_search_svm.best_estimator_

# Evaluierung des optimierten Modells
y_pred = best_svm.predict(X_test)
print(f'Optimierte SVM Genauigkeit: {accuracy_score(y_test, y_pred):.2f}')
print(classification_report(y_test, y_pred))
```

6

9. Modellbereitstellung und Implementierung

Ziel: Implementieren des Modells in der Produktionsumgebung.

Details:

- **Modellbereitstellung:** Bereitstellung des Modells für die Nutzung in der Produktionsumgebung.
- **Integration:** Integration des Modells in bestehende Systeme und Anwendungen.

Vorteile:

- Praktische Anwendung des Modells zur Lösung realer Probleme.
- Skalierbarkeit und Effizienz in der Nutzung.

Beispiel:

```
import joblib

# Speichern des Modells
joblib.dump(best_svm, 'best_svm_model.pkl')

# Laden des Modells (zum Einsatz in der Produktionsumgebung)
loaded_model = joblib.load('best_svm_model.pkl')

# Beispiel für die Vorhersage
example_data = X_test[:5]
predictions = loaded_model.predict(example_data)
print(predictions)
```

10. Überwachung und Wartung

Ziel: Überwachen der Modellleistung und Wartung des Modells.

Details:

- **Modellüberwachung:** Kontinuierliche Überwachung der Modellleistung in der Produktionsumgebung.
- **Modellwartung:** Regelmäßige Updates und Anpassungen des Modells basierend auf neuen Daten und Feedback.

Vorteile:

- Sicherstellung der langfristigen Modellgenauigkeit.
- Anpassung an veränderte Bedingungen und neue Daten.

Beispiel: