3. ESTRUTURA INTERNA DA EMPRESA

3.1. Aprendizado de máquina

3.1.1. Exploração de Dados e Pré-processamento

3.1.1.1. Coleta de Dados relevantes para o negócio proposto pela empresa

```
import pandas as pd
dados = pd.read_csv('/kaggle/input/projetinho/submission.csv')
dados = pd.read_csv('/kaggle/input/projetinho/sample_4E0BhPN.csv')
```

3.1.1.2. Limpeza e Pré-processamento dos Dados

Ausentes

```
import pandas as pd
import numpy as np


data = {
'carro': [Uno, , hb20, np.nan,],
'km': [18, 1000, np.nan ]
}

df = pd.DataFrame(data)

# Preencher valores ausentes com a média
media_carro = df['carro'].mean()
df['idade'] = df['carro'].fillna(media_carro)

# Preencher valores ausentes com a mediana
mediana_carro = df['carro'].median()
df['carro'] = df['carro'].fillna(mediana_salario)

print("DataFrame após preenchimento de valores ausentes:")
print(df)
```

Outliers

```
from scipy import stats

z_scores_km = np.abs(stats.zscore(df['km']))
```

```
limite = 3
outliers_km = df['km'][z_scores_km > limite]

# Substituir outliers pela mediana
mediana_km = df['km'].median()
df['km'] = df['km'].mask(z_scores_km > limite, mediana_km)

print("\nDataFrame após tratamento de outliers no km:")
print(df)
```

Inconsistentes

```
# Verificar dados inconsistentes no km
limite_inferior_km = 18
limite_superior_km = 100
df = df[(df['km'] >= limite_inferior_km) & (df['km'] <= limite_superior_km)]

print("\nDataFrame após remoção de dados inconsistentes no km:")
print(df)
```

3.1.1.3. Verificar a Matriz Confusão

```
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np

# Dados de exemplo
y_true = np.array([0, 0, 1, 1, 0, 1, 0, 0, 0, 1])
y_pred = np.array([1, 0, 0, 1, 0, 1, 1, 1, 1, 1])

# Matriz de confusão
conf_matrix = confusion_matrix(y_true, y_pred)
print("Matriz de Confusão:")
print(conf_matrix)

# Relatório de classificação
print("\nRelatório de Classificação:")
print(classification_report(y_true, y_pred))
```

3.1.2. Implementação de Modelos de Aprendizado de Máquina

3.1.2.1. Escolha de algoritmos de ML Adequados ao Problema

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = clf.predict(X_test)

# Avaliar a precisão do modelo
accuracy = accuracy_score(y_test, y_pred)
print("natureza_dos_dados:", accuracy)
```

3.1.2.2. Implementação dos Modelos Escolhidos Utilizando Bibliotecas como Scikit-learn ou TensorFlow

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Carregar o conjunto de dados Iris
iris = load_iris()
X = iris.data
y = iris.target

# Dividir os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inicializar e treinar o modelo Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = clf.predict(X_test)

# Avaliar o desempenho do modelo
accuracy = accuracy_score(y_test, y_pred)
print("Acurácia do modelo:", accuracy)

# Exibir o relatório de classificação
print("\nRelatório de Classificação:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

3.1.2.3. Avaliação da performance dos modelos com métricas apropriadas

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Carregar o conjunto de dados Iris
iris = load_iris()
X = iris.data
y = iris.target

# Dividir os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inicializar e treinar o modelo Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = clf.predict(X_test)

# Avaliar o desempenho do modelo
report = classification_report(y_test, y_pred, target_names=iris.target_names)

print("Relatório de Classificação:")
print(report)
```

3.1.3. Otimização e Validação do Modelo

3.1.3.1. Otimização dos Hiperparâmetros dos Modelos para Melhorar a Performance

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Carregar o conjunto de dados Iris
iris = load_iris()
X = iris.data
y = iris.target

# Dividir os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Definir os hiperparâmetros para o Grid Search
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Inicializar o classificador RandomForest
rf = RandomForestClassifier(random_state=42)

# Inicializar o Grid Search com validação cruzada
grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='accuracy')

# Realizar o ajuste dos dados de treinamento
grid_search.fit(X_train, y_train)

# Melhores hiperparâmetros encontrados
best_params = grid_search.best_params_
print("Melhores hiperparâmetros encontrados:")
print(best_params)
```

3.1.3.2. Validação Cruzada para Verificar a Robustez do Modelo

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier

# Carregar o conjunto de dados Iris
iris = load_iris()
X = iris.data
y = iris.target

# Inicializar o classificador RandomForest
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Realizar validação cruzada com k-fold (k=5)
cv_scores = cross_val_score(rf, X, y, cv=5)

# Exibir os scores de validação cruzada
print("Scores de validação cruzada:", cv_scores)

# Calcular a média dos scores de validação cruzada
mean_cv_score = cv_scores.mean()
print("Média dos scores de validação cruzada:", mean_cv_score)
```

3.1.3.3. Documentação do Processo de construção e Treinamento do Modelo

Documentação do Processo de Construção e Treinamento do Modelo
Introdução:
Este documento fornece uma visão detalhada do processo de construção e treinamento do modelo de aprendizado de máquina para a tarefa específica. Descreve as etapas, parâmetros selecionados e os resultados obtidos durante o desenvolvimento do modelo.
Objetivo:
O objetivo principal deste modelo é [descrever brevemente o objetivo do modelo, por exemplo, prever vendas futuras, classificar dados, etc.].

Etapas do Processo:
1. Exploração de Dados e Pré-processamento:
Coleta de Dados:
Descreva as fontes de dados utilizadas.
Lista de variáveis/features incluídas.
Limpeza e Pré-processamento:
Identificação e tratamento de valores ausentes, outliers, etc.
Transformações aplicadas aos dados.
2. Implementação de Modelos de Aprendizado de Máquina:
Escolha de Algoritmos:
Justificativa para a escolha dos algoritmos utilizados.
Implementação:
Detalhes sobre como os modelos foram implementados.
Utilização de bibliotecas (por exemplo, Scikit-learn, TensorFlow).
3. Otimização e Validação do Modelo:
Otimização de Hiperparâmetros:
Descrição do processo de otimização.
Lista dos hiperparâmetros ajustados.
Validação Cruzada:
Detalhes sobre como a validação cruzada foi realizada.
Resultados obtidos.
Parâmetros do Modelo:
Lista completa de hiperparâmetros e seus valores finais após a otimização.
Outros parâmetros relevantes para o modelo.
Métricas de Avaliação:
Descrição das métricas utilizadas para avaliar o desempenho do modelo.
Resultados específicos obtidos para cada métrica.

3.2. Ciência de Dados

3.2.1. Análise Descritiva dos Dados

3.2.1.1. Utilização de Técnicas Estatísticas Básicas para descrever os Dados