Predição de Preço de Carros

Rafael P. Rubio

Contexto

Uma empresa de automóveis chinesa deseja entrar no mercado estadunidense de carros, e contratou uma consultoria para entender quais fatores mais influenciam na precificação dos carros de lá.

Dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
car_ID	205.0	103.000000	59.322565	1.00	52.00	103.00	154.00	205.00
symboling	205.0	0.834146	1.245307	-2.00	0.00	1.00	2.00	3.00
wheelbase	205.0	98.756585	6.021776	86.60	94.50	97.00	102.40	120.90
carlength	205.0	174.049268	12.337289	141.10	166.30	173.20	183.10	208.10
carwidth	205.0	65.907805	2.145204	60.30	64.10	65.50	66.90	72.30
carheight	205.0	53.724878	2.443522	47.80	52.00	54.10	55.50	59.80
curbweight	205.0	2555.565854	520.680204	1488.00	2145.00	2414.00	2935.00	4066.00
enginesize	205.0	126.907317	41.642693	61.00	97.00	120.00	141.00	326.00
boreratio	205.0	3.329756	0.270844	2.54	3.15	3.31	3.58	3.94
stroke	205.0	3.255415	0.313597	2.07	3.11	3.29	3.41	4.17
compressionratio	205.0	10.142537	3.972040	7.00	8.60	9.00	9.40	23.00
horsepower	205.0	104.117073	39.544167	48.00	70.00	95.00	116.00	288.00
peakrpm	205.0	5125.121951	476.985643	4150.00	4800.00	5200.00	5500.00	6600.00
citympg	205.0	25.219512	6.542142	13.00	19.00	24.00	30.00	49.00
highwaympg	205.0	30.751220	6.886443	16.00	25.00	30.00	34.00	54.00
price	205.0	13276.710571	7988.852332	5118.00	7788.00	10295.00	16503.00	45400.00

Dataset

```
df.head(5)
```

car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg	price
1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495.0
2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500.0
3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500.0
4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950.0
5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450.0

Análise e Testes

Verificar a correlação entre preço e as demais variáveis

```
df_numerico = df.select_dtypes(include=['float64', 'int64'])
correlacao = df_numerico.corr()['price'].sort_values(ascending=False)
print("Ranking de Influência no Preço:")
print(correlacao)
```

```
Ranking de Influência no Preço:
price               1.000000
enginesize          0.874145
curbweight          0.835305
horsepower          0.808139
carwidth            0.759325
carlength           0.682920
wheelbase           0.577816
boreratio           0.553173
carheight           0.119336
stroke              0.079443
compressionratio    0.067984
symboling          -0.079978
peakrpm            -0.085267
car_ID             -0.109093
citympg            -0.685751
highwaympg         -0.697599
Name: price, dtype: float64
```

Teste 1: Regressão Linear com variáveis selecionadas

```python
def carregar_e_processar_dados(caminho_arquivo="CarPrice_Assignment.csv"):
    try:
        df = pd.read_csv(caminho_arquivo)
        colunas_usadas = ['enginesize', 'curbweight', 'horsepower', 'highwaympg', 'citympg', 'drivewheel', 'price']
        df_modelo = df[colunas_usadas].copy()
        mapeamento_tracao = {'rwd': 0, 'fwd': 1, '4wd': 2}
        df_modelo['drivewheel'] = df_modelo['drivewheel'].map(mapeamento_tracao)

        return df_modelo, df
    except FileNotFoundError:
        return None, None

    return modelo, r2, mae
```

```python
def treinar_modelo(df_modelo):
    X = df_modelo.drop('price', axis=1)
    y = df_modelo['price']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    modelo = LinearRegression()
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    return modelo, r2, mae
```

```python
def prever_preco(modelo, motor, peso, cavalos, consumo_estrada, consumo_cidade, tracao_valor):
    dados_input = pd.DataFrame([[motor, peso, cavalos, consumo_estrada, consumo_cidade, tracao_valor]],
                               columns=['enginesize', 'curbweight', 'horsepower', 'highwaympg', 'citympg', 'drivewheel'])
    return modelo.predict(dados_input)[0]
```

```python
def prever_preco(modelo, motor, peso, cavalos, consumo_estrada, consumo_cidade, tracao_valor):
    dados_input = pd.DataFrame([[motor, peso, cavalos, consumo_estrada, consumo_cidade, tracao_valor]],
                               columns=['enginesize', 'curbweight', 'horsepower', 'highwaympg', 'citympg', 'drivewheel'])
    return modelo.predict(dados_input)[0]

df_limpo, df_original = carregar_e_processar_dados()

if df_limpo is None:
    exit()

print("\n2. Treinando a Inteligência Artificial...")
modelo, r2, mae = treinar_modelo(df_limpo)

print(f"   Modelo treinado!")
print(f"   Precisão (R²): {r2:.2f}")
print(f"   Erro Médio: $ {mae:.2f}")

print("\n3. Simulando um carro para teste...")
motor = 150
peso = 2500
cavalos = 120
consumo_estrada = 30
consumo_cidade = 25
tracao = 0

preco = prever_preco(modelo, motor, peso, cavalos, consumo_estrada, consumo_cidade, tracao)

print(f"   O preço previsto para este carro é: $ {preco:,.2f}")
```

```
2. Treinando a Inteligência Artificial...
   Modelo treinado!
   Precisão (R²): 0.83
   Erro Médio: $ 2660.42

3. Simulando um carro para teste...
   O preço previsto para este carro é: $ 16,354.06
```

Teste 2: Random Forest

Com os mesmos parâmetros, o preço estimado caiu para $11,821.52

A precisão aumentou para mais de 0.90 e o erro está abaixo de 1500

```python
def treinar_modelo(df):
    y = np.log1p(df["price"])
    X = df.drop(columns=["price", "CarName"])
    cat_features = X.select_dtypes(include="object").columns.tolist()
    num_features = X.select_dtypes(exclude="object").columns.tolist()
    preprocessor = ColumnTransformer(
        transformers=[
            ("num", StandardScaler(), num_features),
            ("cat", OneHotEncoder(handle_unknown="ignore"), cat_features)
        ]
    )

    pipeline = Pipeline(
        steps=[
            ("preprocess", preprocessor),
            ("feature_selection", SelectFromModel(
                RandomForestRegressor(n_estimators=100, max_depth=20, random_state=42, n_jobs=-1),
                threshold="median"
            )),
            ("model", RandomForestRegressor(
                n_estimators=300,
                max_depth=25,
                min_samples_split=3,
                min_samples_leaf=1,
                random_state=42,
                n_jobs=-1
            ))
        ]
    )
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
    pipeline.fit(X_train, y_train)
    pred_log = pipeline.predict(X_test)
    pred = np.expm1(pred_log)
    y_test_real = np.expm1(y_test)
    r2 = r2_score(y_test_real, pred)
    mae = mean_absolute_error(y_test_real, pred)

    return pipeline, r2, mae, num_features, cat_features, X_train
```

Modelo escolhido

Foi utilizado um Random Forest Regressor com seleção automática de features (SelectFromModel).

```
pipeline = Pipeline(
    steps=[
        ("preprocess", preprocessor),
        ("feature_selection", SelectFromModel(
            RandomForestRegressor(n_estimators=100, max_depth=20, random_state=42, n_jobs=-1),
            threshold="median"
        )),
        ("model", RandomForestRegressor(
            n_estimators=300,
            max_depth=25,
            min_samples_split=3,
            min_samples_leaf=1,
            random_state=42,
            n_jobs=-1
        ))
    ]
)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
pipeline.fit(X_train, y_train)
pred_log = pipeline.predict(X_test)
pred = np.expm1(pred_log)
y_test_real = np.expm1(y_test)
r2 = r2_score(y_test_real, pred)
mae = mean_absolute_error(y_test_real, pred)
rmse = np.sqrt(mean_squared_error(y_test_real, pred))
mape = np.mean(np.abs((y_test_real - pred) / y_test_real)) * 10
return pipeline, r2, mae, rmse, mape, num_features, cat_features, X_train, y_test_real, pred
```

Modelo escolhido

Também foram criadas novas features para aumentar a precisão

```
df_input["avg_mpg"] = (df_input["citympg"] + df_input["highwaympg"]) / 2
df_input["power_per_engine"] = df_input["horsepower"] / df_input["enginesize"]
df_input["volume"] = df_input["carlength"] * df_input["carwidth"] * df_input["carheight"]
df_input["weight_to_power"] = df_input["curbweight"] / df_input["horsepower"]
```