👋 Hi, I'm **Ashish Mishra**

🎯 Internship in AI/ML

🏢 at ShadowFox

Boston House price prediction

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Ashish Mishra\OneDrive\Desktop\ShadowFox\Shadowfox_AI-M

df.head()
```

Out[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	I
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	

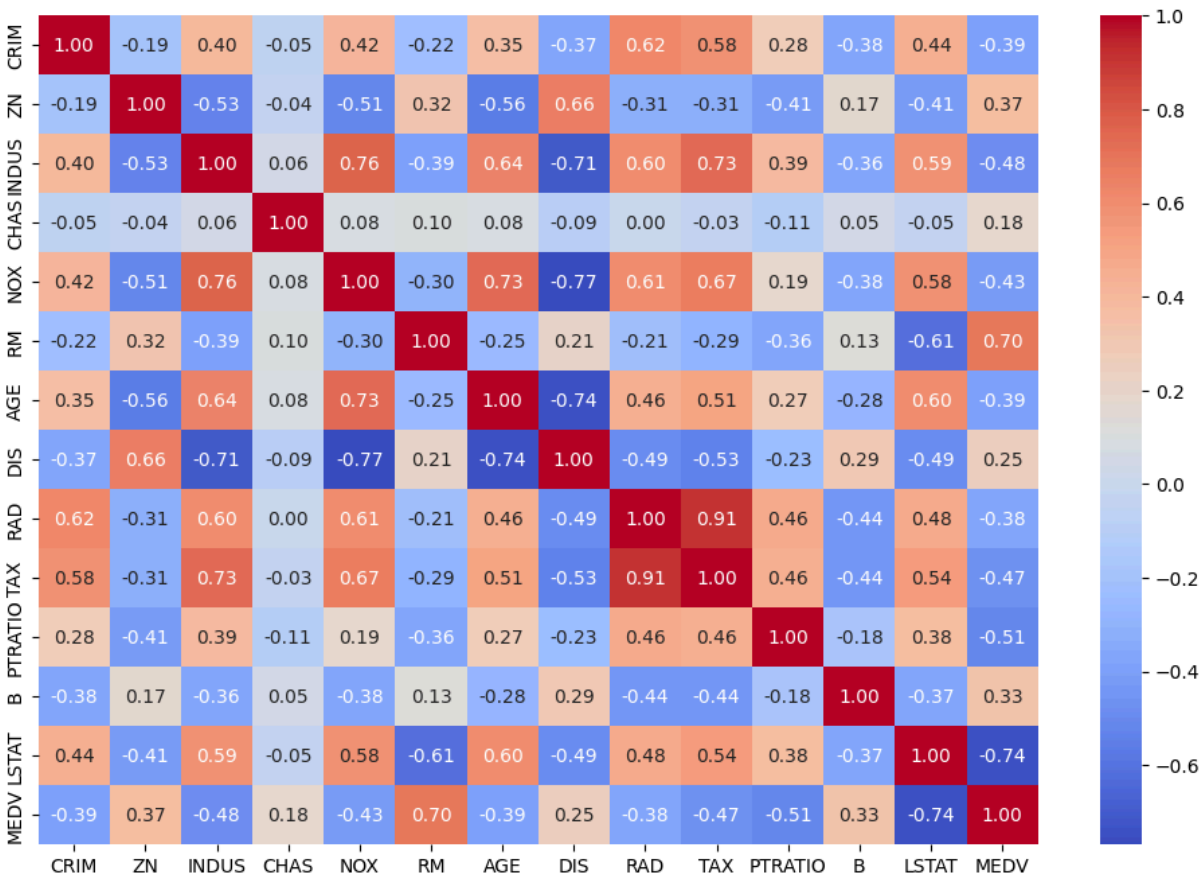
In [3]:
```
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
None
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  486.000000  486.000000  486.000000  486.000000  506.000000  506.000000
mean     3.611874   11.211934   11.083992    0.069959    0.554695    6.284634
std      8.720192   23.388876    6.835896    0.255340    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.081900    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.253715    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.560263   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  486.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.518519    3.795043    9.549407  408.237154   18.455534  356.674032
std     27.999513    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.175000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     76.800000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     93.975000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT        MEDV
count  486.000000  506.000000
mean    12.715432   22.532806
std      7.155871    9.197104
min      1.730000    5.000000
25%      7.125000   17.025000
50%     11.430000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

In [4]:
```
print("Missing values:", df.isnull().sum())
```

```
Missing values: CRIM       20
ZN          20
INDUS       20
CHAS        20
NOX          0
RM           0
AGE         20
DIS          0
RAD          0
TAX          0
PTRATIO      0
B            0
LSTAT       20
MEDV         0
dtype: int64
```

In [5]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.show()
```



In [6]:
```
X = df.drop('MEDV', axis=1)  # Features
y = df['MEDV']  # Target variable
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [8]:
```
imputer = SimpleImputer(strategy='median')
```

In [9]:
```
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
```

In [10]:
```
model = LinearRegression()
model.fit(X_train_imputed, y_train)
```

Out[10]:
```
▾ LinearRegression
LinearRegression()
```

In [11]:
```
y_pred = model.predict(X_test_imputed)
```

In [12]:
```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

In [13]:
```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error (MSE): 24.983445419912112
Root Mean Squared Error (RMSE): 4.998344267846314
R-squared: 0.659318743105165
```

In [14]:
```
plt.figure(figsize=(6, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.show()
```


Actual vs Predicted House Prices
Predicted Prices
Actual Prices
0
10
20
30
40
10
20
30
40
50

👋 Hi, I'm **Ashish Mishra**

🎯 Internship in AI/ML

🏢 at ShadowFox

Car selling price prediction and analysis

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Ashish Mishra\OneDrive\Desktop\ShadowFox\Shadowfox_AI-M
```

In [3]:
```
df.head()
```

Out[3]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Trans
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

In [6]:

```
from datetime import datetime

current_year = datetime.now().year

df['Years_Used'] = current_year - df['Year']
df = df.drop(['Car_Name', 'Year'], axis=1)

df.head()
```

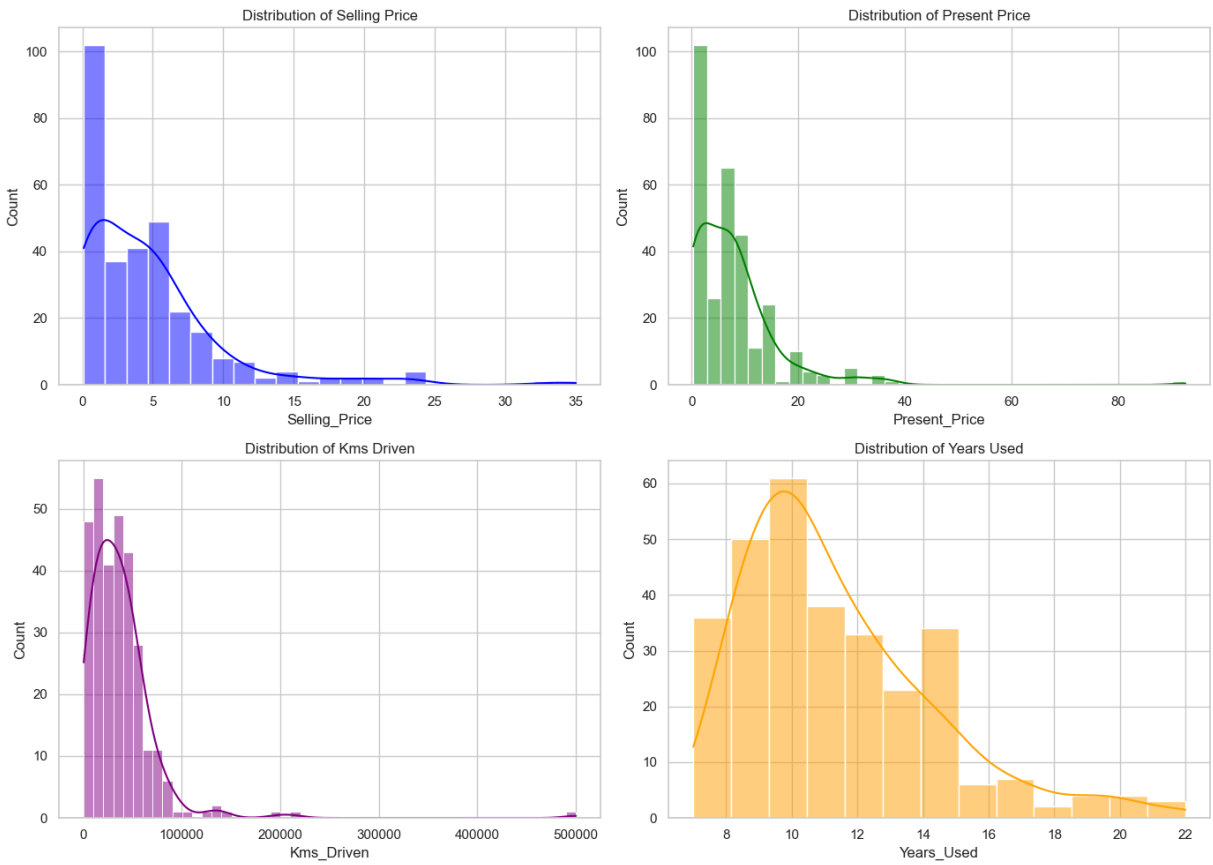
Out[6]:

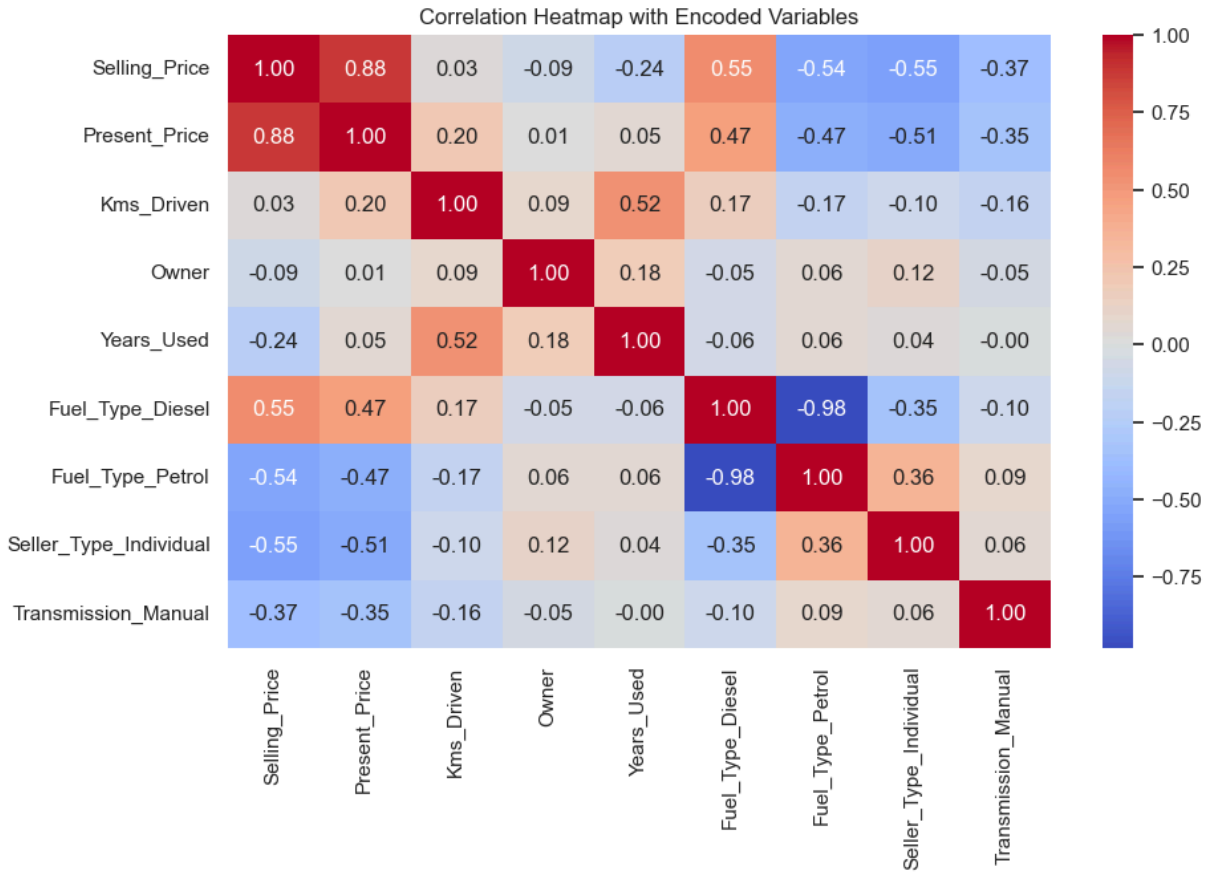
	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	4.60	6.87	42450	Diesel	Dealer	Manual	0

In [ ]:

```
import seaborn as sns

import matplotlib.pyplot as plt

sns.set(style="whitegrid")

fig, axes = plt.subplots(2, 2, figsize=(14, 10))

sns.histplot(df['Selling_Price'], kde=True, ax=axes[0, 0], color="blue")
axes[0, 0].set_title("Distribution of Selling Price")

sns.histplot(df['Present_Price'], kde=True, ax=axes[0, 1], color="green")
axes[0, 1].set_title("Distribution of Present Price")

sns.histplot(df['Kms_Driven'], kde=True, ax=axes[1, 0], color="purple")
axes[1, 0].set_title("Distribution of Kms Driven")
```

```
sns.histplot(df['Years_Used'], kde=True, ax=axes[1, 1], color="orange")
axes[1, 1].set_title("Distribution of Years Used")

plt.tight_layout()
plt.show()

df_encoded = pd.get_dummies(df, columns=['Fuel_Type', 'Seller_Type', 'Transmission'

plt.figure(figsize=(10, 6))
sns.heatmap(df_encoded.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap with Encoded Variables")
plt.show()
```

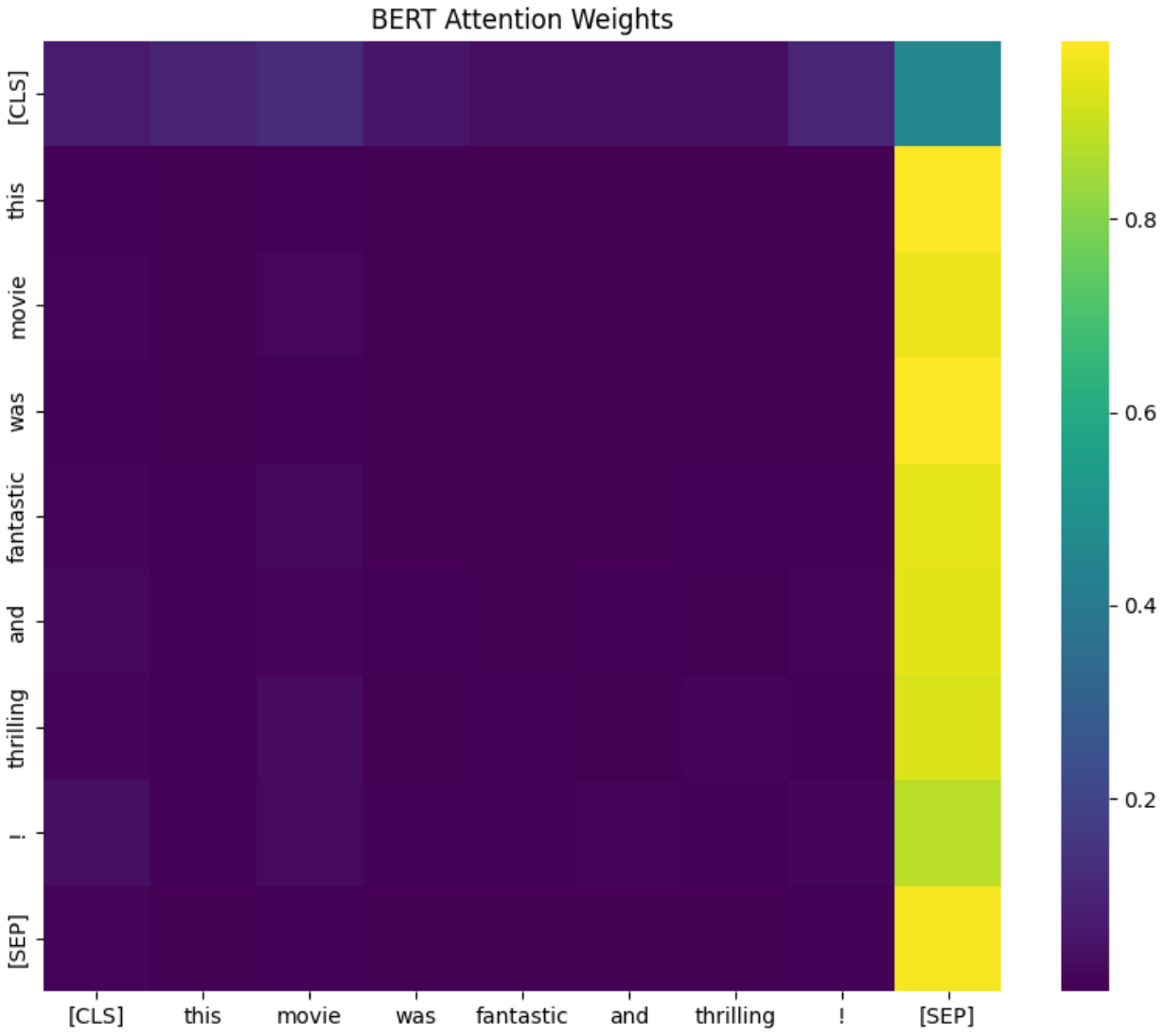




In [ ]:
```
encoded_data = pd.get_dummies(df, columns=['Fuel_Type', 'Seller_Type', 'Transmissio

encoded_data.head()
```

Out[ ]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Years_Used	Fuel_Type_Diesel	Fuel_Typ
0	3.35	5.59	27000	0	11	False	
1	4.75	9.54	43000	0	12	True	
2	7.25	9.85	6900	0	8	False	
3	2.85	4.15	5200	0	14	False	
4	4.60	6.87	42450	0	11	True	


BERT Attention Weights
[CLS]
this
movie
was
fantastic
and
thrilling
!
[SEP]
0.8
0.6
0.4
0.2