car-price-predition-2

December 30, 2023

[39]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

[40]:
```
car = pd.read_csv('/data/notebook_files/car_prediction_data.csv')
```

[41]:
```
car.head()
```

[42]:
```
car.info()
```

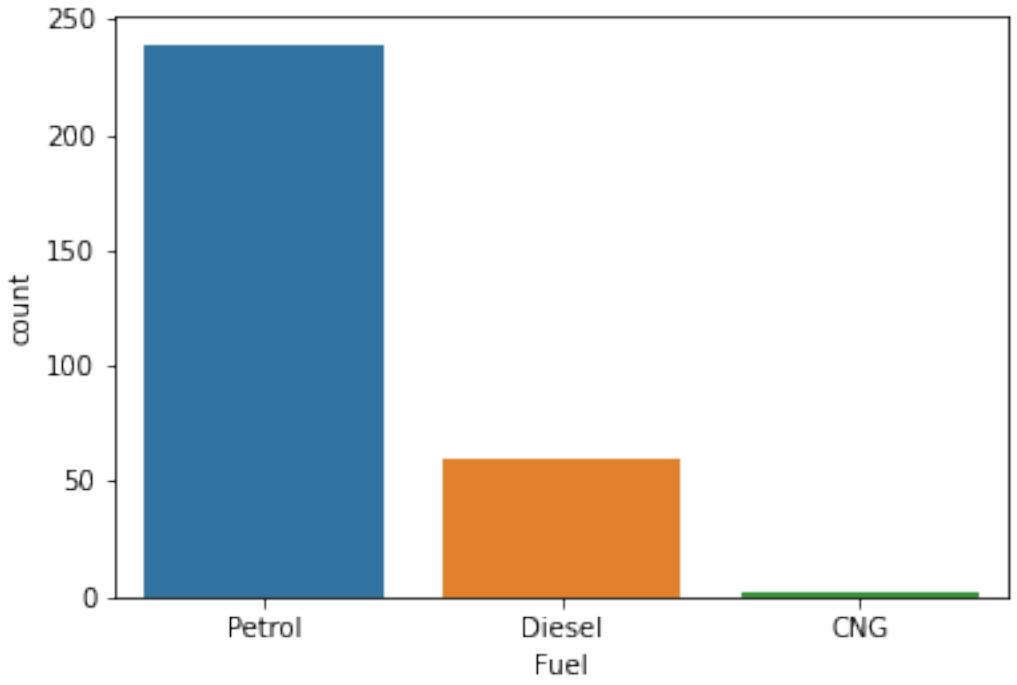
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  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
```

[43]:
```
car.Car_Name.value_counts().reset_index()
```

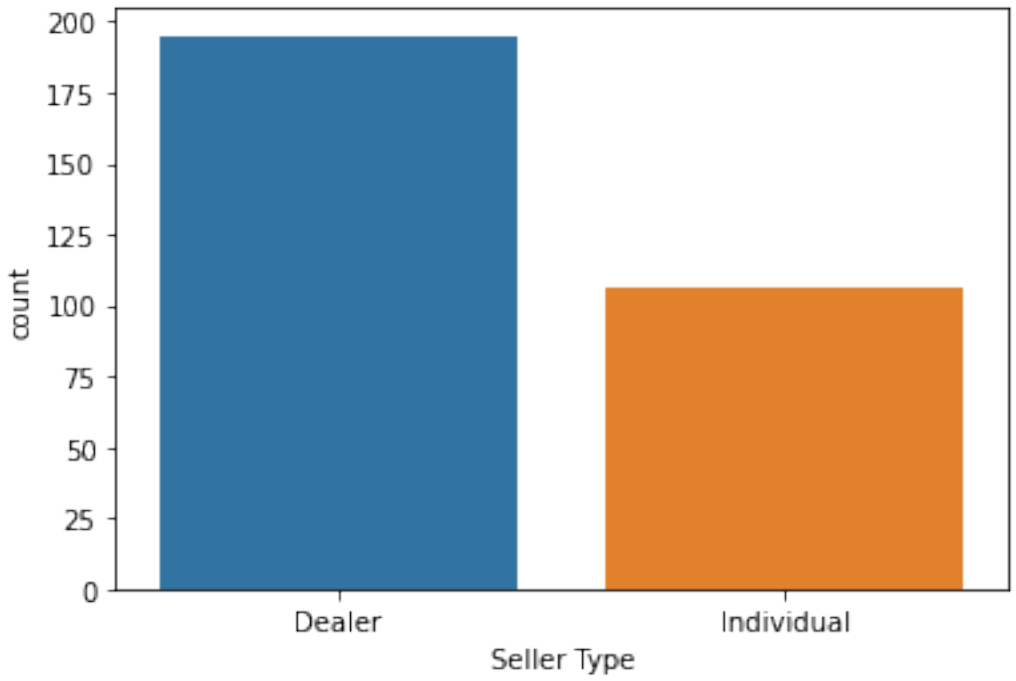
[44]:
```
df_fuel = car.Fuel_Type.value_counts().rename_axis('Fuel').
 ↪reset_index(name='count')
```

[45]:
```
sns.barplot(data=df_fuel,x='Fuel',y='count')
plt.show()
```



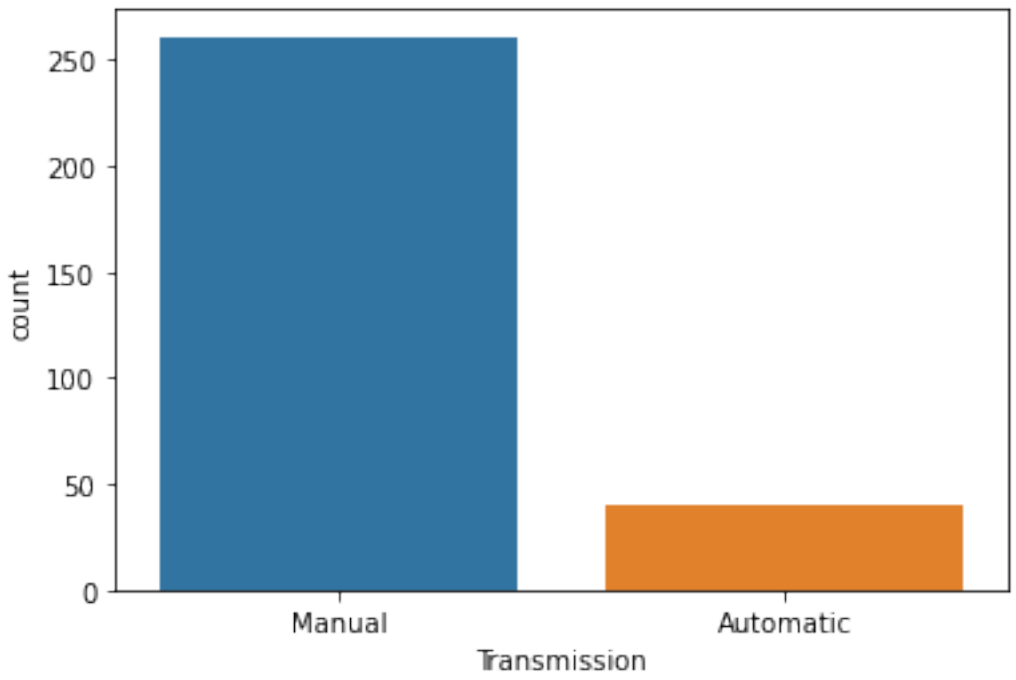
[46]:
```
df_seller = car.Seller_Type.value_counts().rename_axis('Seller Type').
 ↪reset_index(name='count')
```

[47]:
```
sns.barplot(data=df_seller,x='Seller Type',y='count')
plt.show()
```

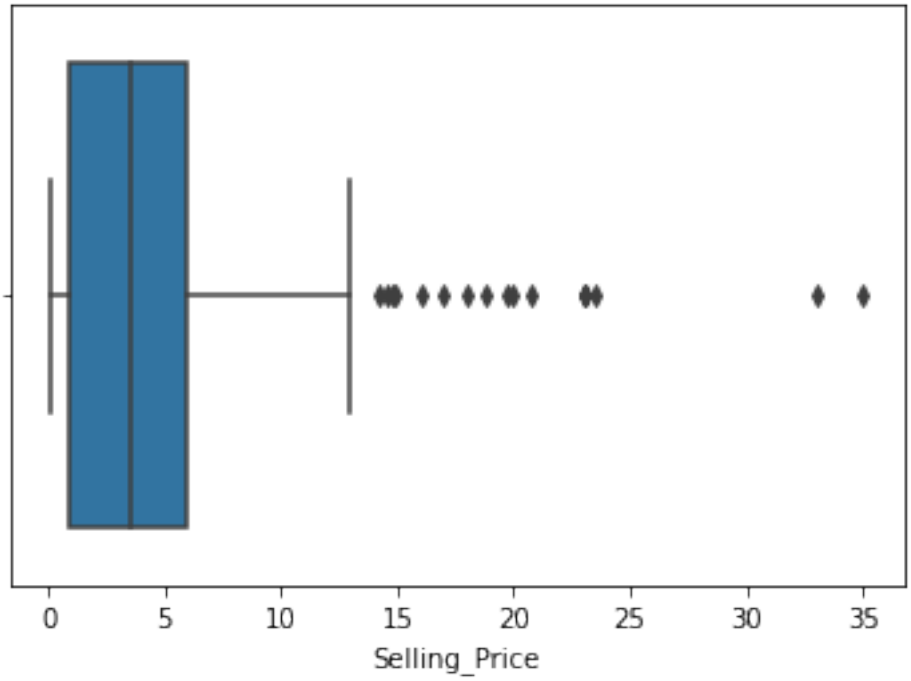


[48]:
```
df_Transmission = car.Transmission.value_counts().rename_axis('Transmission').
↪reset_index(name='count')
```

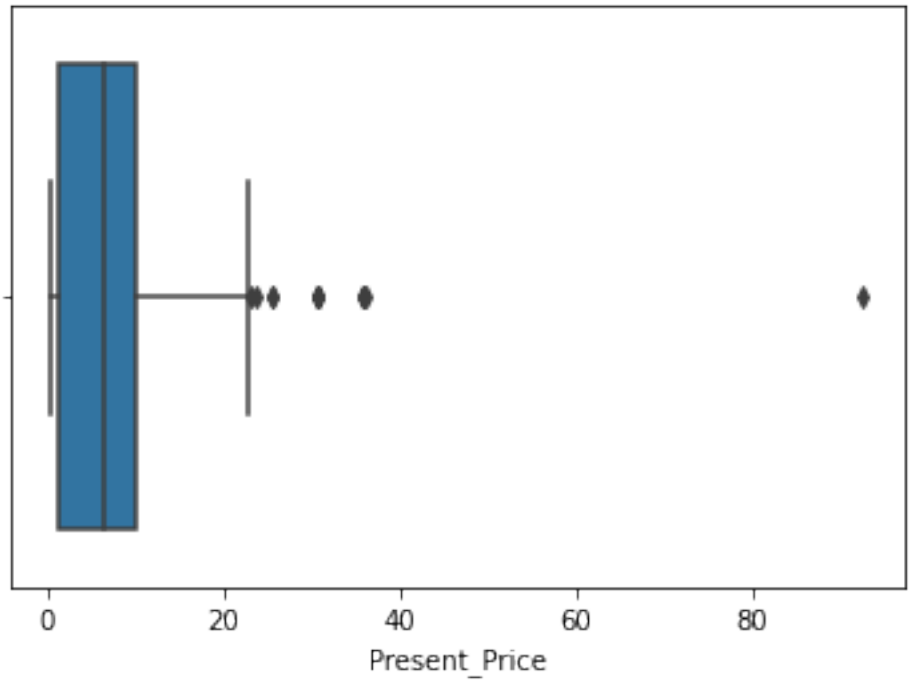
[49]:
```
sns.barplot(data=df_Transmission,x='Transmission',y='count')
plt.show()
```



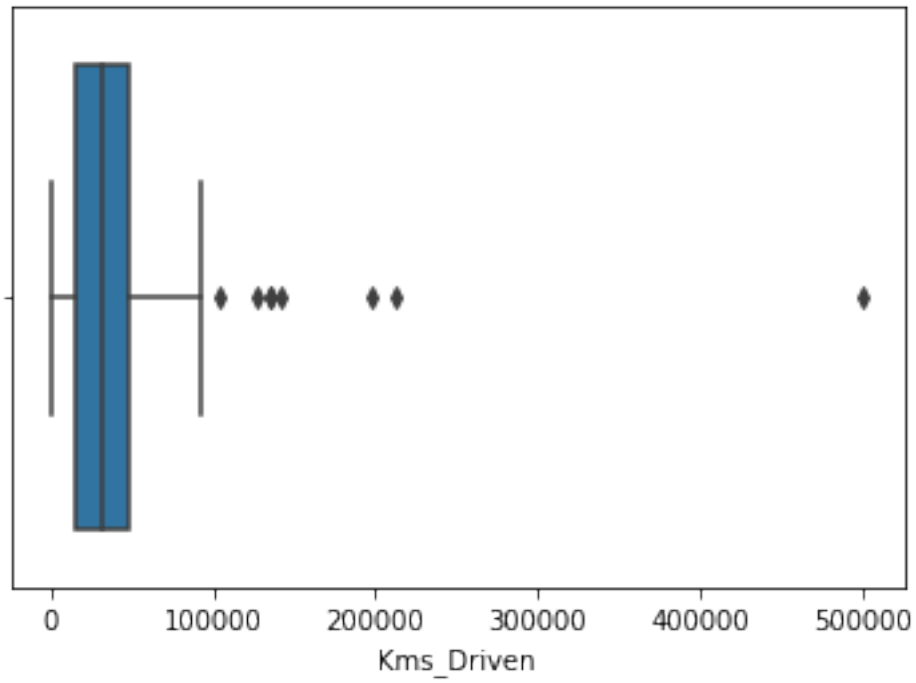
[50]:
```
sns.boxplot(x=car['Selling_Price'])
plt.show()
```



[51]:
```
sns.boxplot(x=car['Present_Price'])
plt.show()
```



[52]:
```
sns.boxplot(x=car['Kms_Driven'])
plt.show()
```



[53]:
```
car.describe()
```

[54]:
```
car.Fuel_Type = LabelEncoder().fit_transform(car.Fuel_Type)
car.Car_Name = LabelEncoder().fit_transform(car.Car_Name)
car.Seller_Type = LabelEncoder().fit_transform(car.Seller_Type)
car.Transmission = LabelEncoder().fit_transform(car.Transmission)
```

[55]:
```
car.head()
```

[56]:
```
car.describe()
```

[57]:
```
car_corr = car.corr()
car_corr['Selling_Price']
```

[58]:
```
car.head()
```

[59]:
```
car.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    int64
 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    int64
 6   Seller_Type    301 non-null    int64
 7   Transmission   301 non-null    int64
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 21.3 KB
```

[60]:
```
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score,KFold
from sklearn.model_selection import train_test_split
```

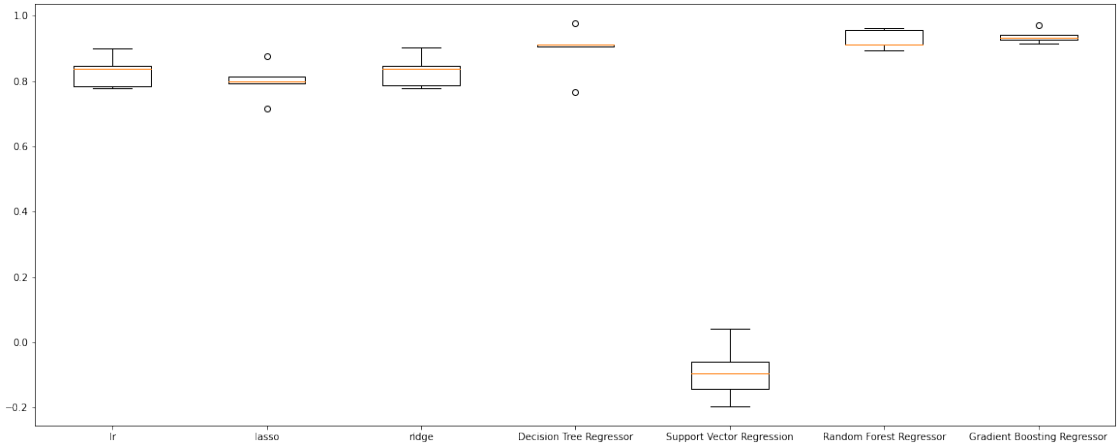
[61]:
```
X = car.drop(['Selling_Price', 'Car_Name'], axis=1)
y = car['Selling_Price']
```

[62]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪2,random_state=42)
```

[63]:
```
models = {'lr':LinearRegression(),'lasso':Lasso(),'ridge':Ridge(),'Decision␣
↪Tree Regressor':DecisionTreeRegressor(),'Support Vector Regression':
↪SVR(),'Random Forest Regressor':RandomForestRegressor(),'Gradient Boosting␣
↪Regressor':GradientBoostingRegressor()}
```

[64]:
```
results = []
for model in models.values():
    kf = KFold(n_splits=5,shuffle=True,random_state=42)
    cv_result = cross_val_score(model,X,y,cv=kf)
    results.append(cv_result)
```

[65]:
```
plt.figure(figsize=(20,8))
plt.boxplot(results, labels=models.keys())
plt.show()
```



[66]:
```
for name, model in models.items():
    model.fit(x_train, y_train)
    test_score = model.score(x_test, y_test)
    print("{} Test Set Accuracy: {}".format(name, test_score))
```

```
lr Test Set Accuracy: 0.8468053957655803
lasso Test Set Accuracy: 0.798551246128469
ridge Test Set Accuracy: 0.8474451929881059
Decision Tree Regressor Test Set Accuracy: 0.9454030727963618
Support Vector Regression Test Set Accuracy: -0.09528098513804473
Random Forest Regressor Test Set Accuracy: 0.9634837339646395
Gradient Boosting Regressor Test Set Accuracy: 0.9721699329913661
```

1 Gradient Boosting Regressor is Best Model