```
import numpy as np
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder,OneHotEncoder
from sklearn.model_selection import train_test_split

df = pd.read_csv('cars.csv')

df.head()
```

	brand	km_driven	fuel	owner	selling_price
0	Maruti	145500	Diesel	First Owner	450000
1	Skoda	120000	Diesel	Second Owner	370000
2	Honda	140000	Petrol	Third Owner	158000
3	Hyundai	127000	Diesel	First Owner	225000
4	Maruti	120000	Petrol	First Owner	130000

```
df['owner'].value_counts()
```

```
First Owner             5289
Second Owner            2105
Third Owner              555
Fourth & Above Owner     174
Test Drive Car             5
Name: owner, dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(

df.drop(columns=['selling_price']),

df['selling_price'],
                                                    test_size=0.2,
                                                    random_state=42
                                                   )

X_train.head()
```

	brand	km_driven	fuel	owner
6518	Tata	2560	Petrol	First Owner
6144	Honda	80000	Petrol	Second Owner
6381	Hyundai	150000	Diesel	Fourth & Above Owner
438	Maruti	120000	Diesel	Second Owner
5939	Maruti	25000	Petrol	First Owner

The Hard Way!

```
# apply ordinal encoder to owner
oe = OrdinalEncoder(categories=[['Test Drive Car', 'Fourth & Above
Owner', 'Third Owner', 'Second Owner', 'First Owner']])

X_train_owner = oe.fit_transform(X_train.loc[:,['owner']])
X_test_owner = oe.transform(X_test.loc[:,['owner']])
```

```
# convert to df
X_train_owner_df =
pd.DataFrame(X_train_owner,columns=oe.get_feature_names_out())
X_test_owner_df =
pd.DataFrame(X_test_owner,columns=oe.get_feature_names_out())

X_train_owner_df.head()
```

```
   owner
0    4.0
1    3.0
2    1.0
3    3.0
4    4.0
```

```
# apply ohe to brand and fuel
ohe = OneHotEncoder(sparse_output=False)

X_train_brand_fuel = ohe.fit_transform(X_train[['brand','fuel']])
X_test_brand_fuel = ohe.transform(X_test[['brand','fuel']])

# converting to dataframe
X_train_brand_fuel_df = pd.DataFrame(X_train_brand_fuel,
columns=ohe.get_feature_names_out())
X_test_brand_fuel_df = pd.DataFrame(X_test_brand_fuel,
columns=ohe.get_feature_names_out())

X_train_brand_fuel_df.head()
```

```
   brand_Ambassador  brand_Ashok  brand_Audi  brand_BMW
brand_Chevrolet  \
0               0.0          0.0         0.0        0.0
0.0
1               0.0          0.0         0.0        0.0
0.0
2               0.0          0.0         0.0        0.0
0.0
3               0.0          0.0         0.0        0.0
0.0
4               0.0          0.0         0.0        0.0
0.0

   brand_Daewoo  brand_Datsun  brand_Fiat  brand_Force
brand_Ford  ...  \
0           0.0           0.0         0.0          0.0
0.0  ...
1           0.0           0.0         0.0          0.0
0.0  ...
2           0.0           0.0         0.0          0.0
0.0  ...
3           0.0           0.0         0.0          0.0
```

```
0.0  ...
4           0.0           0.0          0.0           0.0
0.0  ...

   brand_Renault  brand_Skoda  brand_Tata  brand_Toyota
brand_Volkswagen  \
0            0.0          0.0         1.0           0.0
0.0
1            0.0          0.0         0.0           0.0
0.0
2            0.0          0.0         0.0           0.0
0.0
3            0.0          0.0         0.0           0.0
0.0
4            0.0          0.0         0.0           0.0
0.0

   brand_Volvo  fuel_CNG  fuel_Diesel  fuel_LPG  fuel_Petrol
0          0.0       0.0          0.0       0.0          1.0
1          0.0       0.0          0.0       0.0          1.0
2          0.0       0.0          1.0       0.0          0.0
3          0.0       0.0          1.0       0.0          0.0
4          0.0       0.0          0.0       0.0          1.0

[5 rows x 36 columns]

X_train.head()

        brand  km_driven    fuel                 owner
6518     Tata       2560  Petrol           First Owner
6144    Honda      80000  Petrol          Second Owner
6381  Hyundai     150000  Diesel  Fourth & Above Owner
438    Maruti     120000  Diesel          Second Owner
5939   Maruti      25000  Petrol           First Owner

X_train_rem =
X_train.drop(columns=['brand','fuel','owner'],inplace=True)
X_test_rem =
X_test.drop(columns=['brand','fuel','owner'],inplace=True)

X_train = pd.concat([X_train_rem, X_train_owner_df,
X_train_brand_fuel_df],axis=1)
X_test = pd.concat([X_test_rem, X_test_owner_df,
X_test_brand_fuel_df],axis=1)

X_train.head()

   owner  brand_Ambassador  brand_Ashok  brand_Audi  brand_BMW  \
0    4.0               0.0          0.0         0.0        0.0
1    3.0               0.0          0.0         0.0        0.0
2    1.0               0.0          0.0         0.0        0.0
```

```
3    3.0               0.0          0.0         0.0        0.0
4    4.0               0.0          0.0         0.0        0.0

   brand_Chevrolet  brand_Daewoo  brand_Datsun  brand_Fiat
brand_Force  ...  \
0              0.0           0.0           0.0         0.0
0.0  ...
1              0.0           0.0           0.0         0.0
0.0  ...
2              0.0           0.0           0.0         0.0
0.0  ...
3              0.0           0.0           0.0         0.0
0.0  ...
4              0.0           0.0           0.0         0.0
0.0  ...

   brand_Renault  brand_Skoda  brand_Tata  brand_Toyota
brand_Volkswagen  \
0            0.0          0.0         1.0           0.0
0.0
1            0.0          0.0         0.0           0.0
0.0
2            0.0          0.0         0.0           0.0
0.0
3            0.0          0.0         0.0           0.0
0.0
4            0.0          0.0         0.0           0.0
0.0

   brand_Volvo  fuel_CNG  fuel_Diesel  fuel_LPG  fuel_Petrol
0          0.0       0.0          0.0       0.0          1.0
1          0.0       0.0          0.0       0.0          1.0
2          0.0       0.0          1.0       0.0          0.0
3          0.0       0.0          1.0       0.0          0.0
4          0.0       0.0          0.0       0.0          1.0

[5 rows x 37 columns]
```

The Easy Way!

```
from sklearn.compose import ColumnTransformer

df = pd.read_csv('cars.csv')

X_train, X_test, y_train, y_test = train_test_split(

df.drop(columns=['selling_price']),

df['selling_price'],
                                                    test_size=0.2,
```

```
                                                    random_state=42
                                                )

X_train.head()
```

```
        brand  km_driven    fuel                  owner
6518     Tata       2560  Petrol           First Owner
6144    Honda      80000  Petrol          Second Owner
6381  Hyundai     150000  Diesel  Fourth & Above Owner
438    Maruti     120000  Diesel          Second Owner
5939   Maruti      25000  Petrol           First Owner
```

```
transformer = ColumnTransformer(
    [
        ("ordinal", OrdinalEncoder(categories=[['Test Drive Car',
'Fourth & Above Owner', 'Third Owner', 'Second Owner', 'First
Owner']]), ['owner']),
        ("onehot", OneHotEncoder(sparse_output=False), ['brand',
'fuel'])
    ],
    remainder='passthrough'
)

# setting to get a pandas df
transformer.set_output(transform='pandas')

X_train_transformed = transformer.fit_transform(X_train)
X_test_transformed = transformer.transform(X_test)

transformer.set_output(transform='pandas')

ColumnTransformer(remainder='passthrough',
                  transformers=[('ordinal',
                                 OrdinalEncoder(categories=[['Test
Drive Car',
                                                             'Fourth &
Above '
                                                             'Owner',
                                                             'Third
Owner',
                                                             'Second
Owner',
                                                             'First
Owner']]),
                                 ['owner']),
                                ('onehot',
OneHotEncoder(sparse_output=False),
                                 ['brand', 'fuel'])])

transformer.fit_transform(X_train)
```

```
      ordinal__owner  onehot__brand_Ambassador  onehot__brand_Ashok  \
6518             4.0                       0.0                  0.0
6144             3.0                       0.0                  0.0
6381             1.0                       0.0                  0.0
438              3.0                       0.0                  0.0
5939             4.0                       0.0                  0.0
...              ...                       ...                  ...
5226             4.0                       0.0                  0.0
5390             3.0                       0.0                  0.0
860              4.0                       0.0                  0.0
7603             4.0                       0.0                  0.0
7270             3.0                       0.0                  0.0

      onehot__brand_Audi  onehot__brand_BMW
onehot__brand_Chevrolet  \
6518                 0.0                0.0                       0.0

6144                 0.0                0.0                       0.0

6381                 0.0                0.0                       0.0

438                  0.0                0.0                       0.0

5939                 0.0                0.0                       0.0

...                  ...                ...                       ...

5226                 0.0                0.0                       0.0

5390                 0.0                0.0                       0.0

860                  0.0                0.0                       0.0

7603                 0.0                0.0                       0.0

7270                 0.0                0.0                       0.0


      onehot__brand_Daewoo  onehot__brand_Datsun
onehot__brand_Fiat  \
6518                   0.0                   0.0                  0.0

6144                   0.0                   0.0                  0.0

6381                   0.0                   0.0                  0.0

438                    0.0                   0.0                  0.0

5939                   0.0                   0.0                  0.0

...                    ...                   ...                  ...
```

```
5226                     0.0                     0.0                     0.0

5390                     0.0                     0.0                     0.0

860                      0.0                     0.0                     0.0

7603                     0.0                     0.0                     0.0

7270                     0.0                     0.0                     0.0


      onehot__brand_Force  ...  onehot__brand_Skoda
onehot__brand_Tata  \
6518                   0.0  ...                   0.0
1.0
6144                   0.0  ...                   0.0
0.0
6381                   0.0  ...                   0.0
0.0
438                    0.0  ...                   0.0
0.0
5939                   0.0  ...                   0.0
0.0
...                    ...  ...                   ...                   ..
.
5226                   0.0  ...                   0.0
0.0
5390                   0.0  ...                   0.0
0.0
860                    0.0  ...                   0.0
0.0
7603                   0.0  ...                   0.0
0.0
7270                   0.0  ...                   0.0
0.0

      onehot__brand_Toyota  onehot__brand_Volkswagen
onehot__brand_Volvo  \
6518                   0.0                       0.0
0.0
6144                   0.0                       0.0
0.0
6381                   0.0                       0.0
0.0
438                    0.0                       0.0
0.0
5939                   0.0                       0.0
0.0
...                    ...                       ...
```

```
...
5226                    0.0                         0.0
0.0
5390                    0.0                         0.0
0.0
860                     0.0                         0.0
0.0
7603                    0.0                         0.0
0.0
7270                    0.0                         0.0
0.0

      onehot__fuel_CNG  onehot__fuel_Diesel  onehot__fuel_LPG  \
6518               0.0                  0.0               0.0
6144               0.0                  0.0               0.0
6381               0.0                  1.0               0.0
438                0.0                  1.0               0.0
5939               0.0                  0.0               0.0
...                ...                  ...               ...
5226               0.0                  1.0               0.0
5390               0.0                  1.0               0.0
860                0.0                  0.0               0.0
7603               0.0                  1.0               0.0
7270               0.0                  0.0               0.0

      onehot__fuel_Petrol  remainder__km_driven
6518                  1.0                  2560
6144                  1.0                 80000
6381                  0.0                150000
438                   0.0                120000
5939                  1.0                 25000
...                   ...                   ...
5226                  0.0                120000
5390                  0.0                 80000
860                   1.0                 35000
7603                  0.0                 27000
7270                  1.0                 70000

[6502 rows x 38 columns]

transformer.feature_names_in_

array(['brand', 'km_driven', 'fuel', 'owner'], dtype=object)

transformer.get_feature_names_out()

array(['ordinal__owner', 'onehot__brand_Ambassador',
       'onehot__brand_Ashok', 'onehot__brand_Audi',
'onehot__brand_BMW',
       'onehot__brand_Chevrolet', 'onehot__brand_Daewoo',
```

```
       'onehot__brand_Datsun', 'onehot__brand_Fiat',
       'onehot__brand_Force', 'onehot__brand_Ford',
'onehot__brand_Honda',
       'onehot__brand_Hyundai', 'onehot__brand_Isuzu',
       'onehot__brand_Jaguar', 'onehot__brand_Jeep',
'onehot__brand_Kia',
       'onehot__brand_Land', 'onehot__brand_Lexus',
'onehot__brand_MG',
       'onehot__brand_Mahindra', 'onehot__brand_Maruti',
       'onehot__brand_Mercedes-Benz', 'onehot__brand_Mitsubishi',
       'onehot__brand_Nissan', 'onehot__brand_Opel',
       'onehot__brand_Peugeot', 'onehot__brand_Renault',
       'onehot__brand_Skoda', 'onehot__brand_Tata',
       'onehot__brand_Toyota', 'onehot__brand_Volkswagen',
       'onehot__brand_Volvo', 'onehot__fuel_CNG',
'onehot__fuel_Diesel',
       'onehot__fuel_LPG', 'onehot__fuel_Petrol',
'remainder__km_driven'],
      dtype=object)
```

```
transformer.n_features_in_
```

```
4
```

```
transformer.transformers_
```

```
[('ordinal',
  OrdinalEncoder(categories=[['Test Drive Car', 'Fourth & Above
Owner',
                              'Third Owner', 'Second Owner', 'First
Owner']]),
  ['owner']),
 ('onehot', OneHotEncoder(sparse_output=False), ['brand', 'fuel']),
 ('remainder', 'passthrough', [1])]
```

```
transformer.output_indices_
```

```
{'ordinal': slice(0, 1, None),
 'onehot': slice(1, 37, None),
 'remainder': slice(37, 38, None)}
```

Sklearn Pipeline

```
df = pd.read_csv('cars.csv')
df.head()
```

```
     brand  km_driven    fuel          owner  selling_price
0   Maruti     145500  Diesel   First Owner         450000
1    Skoda     120000  Diesel  Second Owner         370000
2    Honda     140000  Petrol   Third Owner         158000
```

```
3  Hyundai     127000  Diesel   First Owner         225000
4   Maruti     120000  Petrol   First Owner         130000
```

```
df.shape
```

```
(8128, 5)
```

```
import numpy as np

np.random.seed(42)
missing_km_indices = np.random.choice(df.index,
size=int(0.05*len(df)), replace=False)
df.loc[missing_km_indices, 'km_driven'] = np.nan

# Introduce missing values in 'owner' column (1% missing values)
missing_owner_indices = np.random.choice(df.index,
size=int(0.01*len(df)), replace=False)
df.loc[missing_owner_indices, 'owner'] = np.nan


df.isnull().sum()
```

```
brand              0
km_driven        406
fuel               0
owner             81
selling_price      0
dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(

df.drop(columns=['selling_price']),

df['selling_price'],
                                                    test_size=0.2,
                                                    random_state=42
                                                  )

X_train.head()
```

```
        brand  km_driven    fuel                  owner
6518     Tata     2560.0  Petrol            First Owner
6144    Honda    80000.0  Petrol           Second Owner
6381  Hyundai   150000.0  Diesel  Fourth & Above Owner
438    Maruti   120000.0  Diesel           Second Owner
5939   Maruti    25000.0  Petrol            First Owner
```

```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6502 entries, 6518 to 7270
Data columns (total 4 columns):
```

```
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   brand      6502 non-null   object 
 1   km_driven  6502 non-null   float64
 2   fuel       6502 non-null   object 
 3   owner      6442 non-null   object 
dtypes: float64(1), object(3)
memory usage: 254.0+ KB
```

```
# Plan of Attack

# Missing value imputation
# Encoding Categorical Variables
# Scaling
# Feature Selection
# Model building
# Prediction
```

```
df['owner'].value_counts()
```

```
First Owner             5235
Second Owner            2085
Third Owner              549
Fourth & Above Owner     173
Test Drive Car             5
Name: owner, dtype: int64
```

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest,chi2

from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression

# imputation transformer
trf1 = ColumnTransformer([
    ('impute_km_driven',SimpleImputer(),[1]),
    ('impute_owner',SimpleImputer(strategy='most_frequent'),[3])
],remainder='passthrough')

# encoding categorical variables
trf2 = ColumnTransformer(
    [
        ("ordinal", OrdinalEncoder(handle_unknown='use_encoded_value',
unknown_value=-1), [3]),
        ("onehot", OneHotEncoder(handle_unknown='ignore',
sparse_output=False), [0,2])
    ],
    remainder='passthrough'
)
```

```
# Scaling
trf3 = ColumnTransformer([
    ('scale',MinMaxScaler(),slice(0,38))
])

a = [1,2,3,4,5]
x = slice(0,5)
a[x]
```

```
[1, 2, 3, 4, 5]
```

```
# Feature selection
trf4 = SelectKBest(score_func=chi2,k=10)

# train the model
trf5 = RandomForestRegressor()

from sklearn.pipeline import Pipeline

pipe = Pipeline([
    ('imputer',trf1),
    ('encoder',trf2),
    ('scaler',trf3),
    ('fselector',trf4),
    ('model',trf5)
])

pipe.fit(X_train, y_train)
```

```
Pipeline(steps=[('imputer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('impute_km_driven',
                                                  SimpleImputer(),
[1]),
                                                 ('impute_owner',

SimpleImputer(strategy='most_frequent'),
                                                  [3])])),
                ('encoder',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ordinal',

OrdinalEncoder(handle_unknown='use_encoded_value',

unknown_value=-1),
                                                  [3]),
                                                 ('onehot',

OneHotEncoder(handle_unknown='ignore',

sparse_output=False),
```

```
                                                  [0, 2])])),
                ('scaler',
                 ColumnTransformer(transformers=[('scale',
MinMaxScaler(),
                                                  slice(0, 38,
None))])),
                ('fselector',
                 SelectKBest(score_func=<function chi2 at
0x7fe5a1b6ea70>)),
                ('model', RandomForestRegressor())])
```

```
pipe.feature_names_in_
```

```
array(['brand', 'km_driven', 'fuel', 'owner'], dtype=object)
```

```
pipe.named_steps
```

```
{'imputer': ColumnTransformer(remainder='passthrough',
                  transformers=[('impute_km_driven', SimpleImputer(),
[1]),
                                ('impute_owner',

SimpleImputer(strategy='most_frequent'),
                                 [3])]),
 'encoder': ColumnTransformer(remainder='passthrough',
                  transformers=[('ordinal', OrdinalEncoder(), [3]),
                                ('onehot',
OneHotEncoder(sparse_output=False),
                                 [0, 2])]),
 'scaler': ColumnTransformer(transformers=[('scale', MinMaxScaler(),
slice(0, 37, None))]),
 'fselector': SelectKBest(score_func=<function chi2 at
0x7fe5a1b6ea70>),
 'model': RandomForestRegressor()}
```

```
pipe.named_steps['scaler'].transformers_[0][1].data_max_
```

```
array([3., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1.])
```

```
pipe.predict(X_test)[10:40]
```

```
array([771854.48794078, 771854.48794078, 441301.94073954,
771854.48794078,
       771854.48794078, 771854.48794078, 441301.94073954,
441301.94073954,
       771854.48794078, 441301.94073954, 771854.48794078,
441301.94073954,
```

```
       771854.48794078, 937130.76154139, 441301.94073954,
771854.48794078,
       771854.48794078, 771854.48794078, 771854.48794078,
441301.94073954,
       771854.48794078, 606578.21434016, 441301.94073954,
771854.48794078,
       771854.48794078, 771854.48794078, 441301.94073954,
771854.48794078,
       771854.48794078, 441301.94073954])
```

```
# Predict
pipe.predict(np.array(['Maruti',100000.0,'Diesel','First
Owner']).reshape(1,4))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but SimpleImputer
was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but SimpleImputer
was fitted with feature names
  warnings.warn(

array([630635.02066264])
```

Cross Validation

```
# cross validation using cross_val_score
from sklearn.model_selection import cross_val_score
cross_val_score(pipe, X_train, y_train, cv=5,
scoring='neg_mean_squared_error').mean()
```

```
-639139970114.0674
```

Hyperparameter Tuning

```
# gridsearchcv
params = {
    'model__max_depth':[1,2,3,4,5,None]
}

from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(pipe, params, cv=5,
scoring='neg_mean_squared_error')
grid.fit(X_train, y_train)
```

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('imputer',

ColumnTransformer(remainder='passthrough',
```

```
transformers=[('impute_km_driven',

SimpleImputer(),

[1]),

('impute_owner',

SimpleImputer(strategy='most_frequent'),

[3])])),
                                        ('encoder',

ColumnTransformer(remainder='passthrough',

transformers=[('ordinal',

OrdinalEncoder(handle_unknown='use_encoded_value',

unknow...

('onehot',

OneHotEncoder(handle_unknown='ignore',

sparse_output=False),

[0,

2])])),
                                        ('scaler',

ColumnTransformer(transformers=[('scale',

MinMaxScaler(),

slice(0, 38, None))])),
                                        ('fselector',

SelectKBest(score_func=<function chi2 at 0x7fe5a1b6ea70>)),
                                        ('model',
RandomForestRegressor())]),
             param_grid={'model__max_depth': [1, 2, 3, 4, 5, None]},
             scoring='neg_mean_squared_error')

grid.best_score_

-639082881304.7794

grid.best_params_
```

```
{'model__max_depth': None}
```

Export the Pipeline

```
# export
import pickle
pickle.dump(pipe,open('pipe.pkl','wb'))
```