Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский государственный технический университет имени Н.Э. Баумана» (МГТУ им. Н.Э. Баумана)

Лабораторная работа 5
по курсу «Технологии машинного обучения»

Выполнил
студент группы ИУ5-64
XXX

Москва, 2021

Цель работы

изучение ансамблей моделей машинного обучения.

Задание

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Ход работы

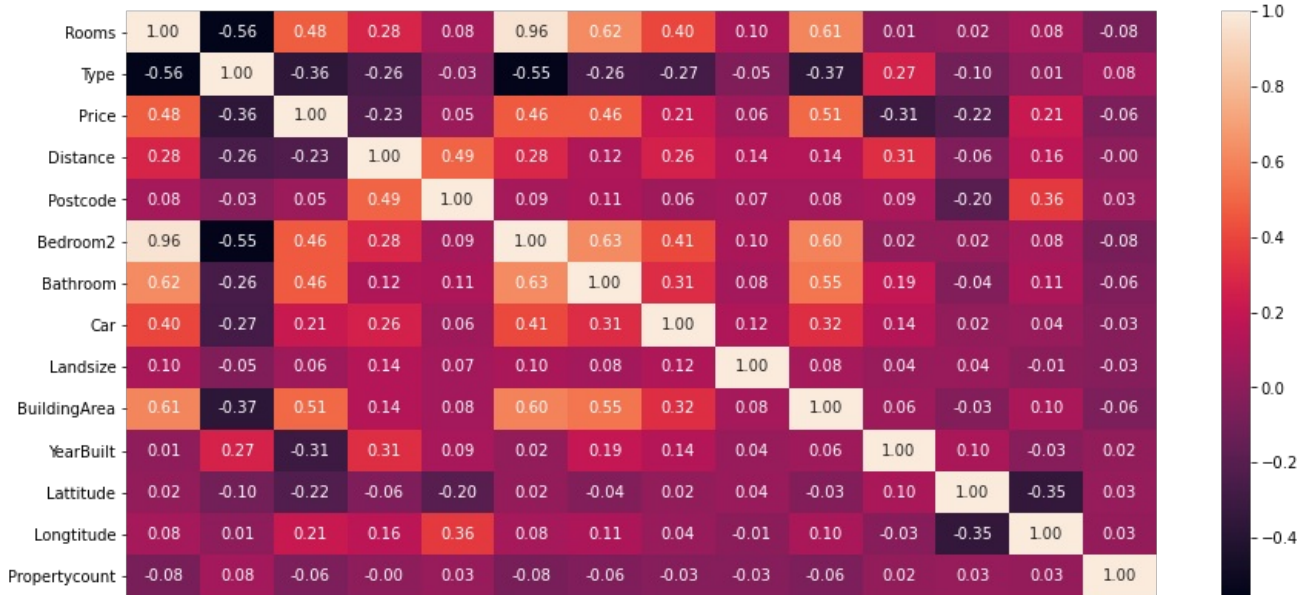
In [1]:
```
# Загрузка данных
import pandas as pd
data = pd.read_csv("../2/melbourne_housing.csv")
```

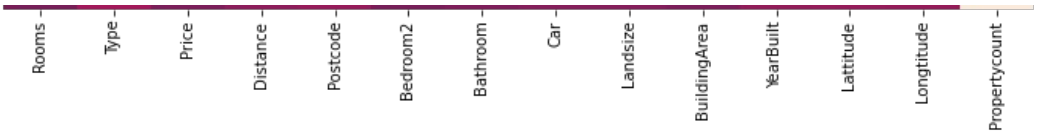
In [2]:
```
import numpy as np
from sklearn.preprocessing import LabelEncoder

columns_and_types = {
  "Rooms": np.int64,
  "Type": None,
  "Price": np.int64,
  "Distance": np.float64,
  "Postcode": np.int64,
  "Bedroom2": np.int64,
  "Bathroom": np.int64,
  "Car": np.int64,
  "Landsize": np.float64,
  "BuildingArea": np.float64,
  "YearBuilt": np.int64,
  "Lattitude": np.float64,
  "Longtitude": np.float64,
  "Propertycount": np.int64,
}

data = data[list(columns_and_types.keys())]
data.dropna(axis=0, how='any', inplace=True)
data = data.astype({k: v for k,v in columns_and_types.items() if v is not None})

type_encoder = LabelEncoder()
data["Type"] = type_encoder.fit_transform(data["Type"])
```

In [3]:
```
from sklearn.preprocessing import MinMaxScaler

sc2 = MinMaxScaler()
for col in data.columns:
  if col != "Price":
    data[col] = sc2.fit_transform(data[[col]])

display(data.head())
```

	Rooms	Type	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Lattitude	Longtitude	Property
2	0.090909	0.0	1035000	0.052743	0.068577	0.166667	0.000	0.0	0.003645	0.025386	0.855407	0.477684	0.516625	0.
4	0.181818	0.0	1465000	0.052743	0.068577	0.250000	0.125	0.0	0.003131	0.048201	0.855407	0.475859	0.517532	0.
6	0.272727	0.0	1600000	0.052743	0.068577	0.250000	0.000	0.2	0.002804	0.045630	0.993925	0.478596	0.517260	0.
11	0.181818	0.0	1876000	0.052743	0.068577	0.333333	0.125	0.0	0.005724	0.067481	0.867558	0.484853	0.521976	0.
14	0.090909	0.0	1636000	0.052743	0.068577	0.166667	0.000	0.2	0.005981	0.034383	0.843256	0.480161	0.518439	0.

In [4]:
```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data.corr(method="pearson"), ax=ax, annot=True, fmt='.2f');
```





In [5]:
```
from sklearn.model_selection import train_test_split

data_X = data.drop("Price", axis=1)
data_Y = data.loc[:, "Price"]
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(
  data_X,
  data_Y,
  test_size=0.2,
  random_state=1
)
```

In [6]:
```
from sklearn.ensemble import BaggingRegressor

bc1 = BaggingRegressor(
  n_estimators=30,
  oob_score=True,
  random_state=10,
)
bc1.fit(data_X_train, data_y_train)
```

Out[6]:
```
BaggingRegressor(n_estimators=30, oob_score=True, random_state=10)
```

In [7]:
```
from sklearn.metrics import r2_score, mean_absolute_error

bc1_y_test_predict = bc1.predict(data_X_test)
r2_score(data_y_test, bc1_y_test_predict), mean_absolute_error(data_y_test, bc1_y_test_predict)
```

Out[7]:
```
(0.8106264070965494, 172662.84650668665)
```

In [8]:
```
from sklearn.ensemble import RandomForestRegressor

tree1 = RandomForestRegressor(
  n_estimators=30,
  oob_score=True,
  random_state=10,
)
tree1.fit(data_X_train, data_y_train)
```

Out[8]:
```
RandomForestRegressor(n_estimators=30, oob_score=True, random_state=10)
```

In [9]:
```
tree1_y_test_predict = tree1.predict(data_X_test)
r2_score(data_y_test, tree1_y_test_predict), \
  mean_absolute_error(data_y_test, tree1_y_test_predict)
```

Out[9]:
```
(0.8141005221956045, 172426.21709786277)
```

In [10]:
```
from sklearn.ensemble import GradientBoostingRegressor

gb1 = GradientBoostingRegressor(random_state=10)
gb1.fit(data_X_train, data_y_train)
```

Out[10]:
```
GradientBoostingRegressor(random_state=10)
```

In [11]:
```
gb1_y_test_predict = gb1.predict(data_X_test)
r2_score(data_y_test, gb1_y_test_predict), \
  mean_absolute_error(data_y_test, gb1_y_test_predict)
```

Out[11]:
```
(0.7962475208768822, 177998.40616463215)
```

In [ ]: