20250609_01

June 9, 2025

[1]:
```
import pandas as pd

data = pd.read_csv('melb_data.csv')
data.head()
```

[1]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	\
0	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	
3	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	
4	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	

	Date	Distance	Postcode	…	Bathroom	Car	Landsize	BuildingArea	\
0	3/12/2016	2.5	3067.0	…	1.0	1.0	202.0	NaN	
1	4/02/2016	2.5	3067.0	…	1.0	0.0	156.0	79.0	
2	4/03/2017	2.5	3067.0	…	2.0	0.0	134.0	150.0	
3	4/03/2017	2.5	3067.0	…	2.0	1.0	94.0	NaN	
4	4/06/2016	2.5	3067.0	…	1.0	2.0	120.0	142.0	

	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	\
0	NaN	Yarra	-37.7996	144.9984	Northern Metropolitan	
1	1900.0	Yarra	-37.8079	144.9934	Northern Metropolitan	
2	1900.0	Yarra	-37.8093	144.9944	Northern Metropolitan	
3	NaN	Yarra	-37.7969	144.9969	Northern Metropolitan	
4	2014.0	Yarra	-37.8072	144.9941	Northern Metropolitan	

	Propertycount
0	4019.0
1	4019.0
2	4019.0
3	4019.0
4	4019.0

[5 rows x 21 columns]

[3]:
```
# Handle missing values like what I learnt yesterday.
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
```

```
X = data.drop('Price', axis = 1)
y = data['Price']

num_cols = X.select_dtypes(include = ['int64', 'float64']).columns
cat_cols = X.select_dtypes(include = ['object']).columns

# Handle categorical cols
high_card_cols = [col for col in cat_cols if X[col].nunique() > 50]
cat_cols = [col for col in cat_cols if col not in high_card_cols]
X_cat = pd.get_dummies(X[cat_cols], dummy_na = True)

# Handle numerical cols
imputer_num = SimpleImputer(strategy = 'mean')
X_num = pd.DataFrame(imputer_num.fit_transform(X[num_cols]), columns = num_cols)

X_processed = pd.concat([X_num, X_cat], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X_processed, y,␣
 ↪random_state = 42)
```

[11]:
```
# Now we try random forest
from sklearn.ensemble import RandomForestRegressor

# n_estimators means how many decision trees are in your random forest, and␣
 ↪it's defaulted to 100.
model_rf = RandomForestRegressor(n_estimators = 150, random_state = 42)

model_rf.fit(X_train, y_train)

# Evaluate
print("Random Forest R² Score:", model_rf.score(X_test, y_test))
```

```
Random Forest R² Score: 0.8116026274961113
```

[17]:
```
# Now we try gradient boosting
# Gradient boosting basically means that we put many 'weak' model together to␣
 ↪make the result stonger.
from sklearn.ensemble import GradientBoostingRegressor

# n_estimators means how many decision trees are in your random forest, and␣
 ↪it's defaulted to 100.
# learning rate means How much each tree contributes to the correction, and␣
 ↪it's defaulted to 0.1.
# although didn't show here, but max_depth is defaulted to 3.
model_gb = GradientBoostingRegressor(n_estimators = 150, learning_rate = 0.2,␣
 ↪random_state = 42)
```

```
model_gb.fit(X_train, y_train)

# Evaluate
print("Gradient Boosting R² Score:", model_gb.score(X_test, y_test))
```

```
Gradient Boosting R² Score: 0.8198181473564012
```

[20]:
```
# Now try put everything into a pipeline
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

# Pipeline: Imputer + Model
pipeline = Pipeline([('model', GradientBoostingRegressor(random_state = 42))])

# Grid of hyperparameters
param_grid = {'model__n_estimators': [100, 150, 200],
              'model__learning_rate': [0.05, 0.1, 0.2],
              'model__max_depth': [3, 5, 7]}

# Setup Grid Search
# n_jobs = -1 is used to fasten the process.
grid = GridSearchCV(pipeline, param_grid, cv = 5, scoring = 'r2', n_jobs = -1)

# Fit on full training set
grid.fit(X_train, y_train)

# Evaluate on test set
print("Best R² score from CV:", grid.best_score_)
print("Test R² score from best model:", grid.best_estimator_.score(X_test,␣
 ↪y_test))
print("Best params:", grid.best_params_)
```

```
Best R² score from CV: 0.8035104901213511
Test R² score from best model: 0.8306703009799219
Best params: {'model__learning_rate': 0.05, 'model__max_depth': 7,
'model__n_estimators': 200}
```