Capstone Project : Prediksi Harga Rumah di King County, USA menggunakan Xgboost Regressor

Rizky Alif Ramadhan (DAI-006) [Universitas Gadjah Mada]

Tools yang Digunakan

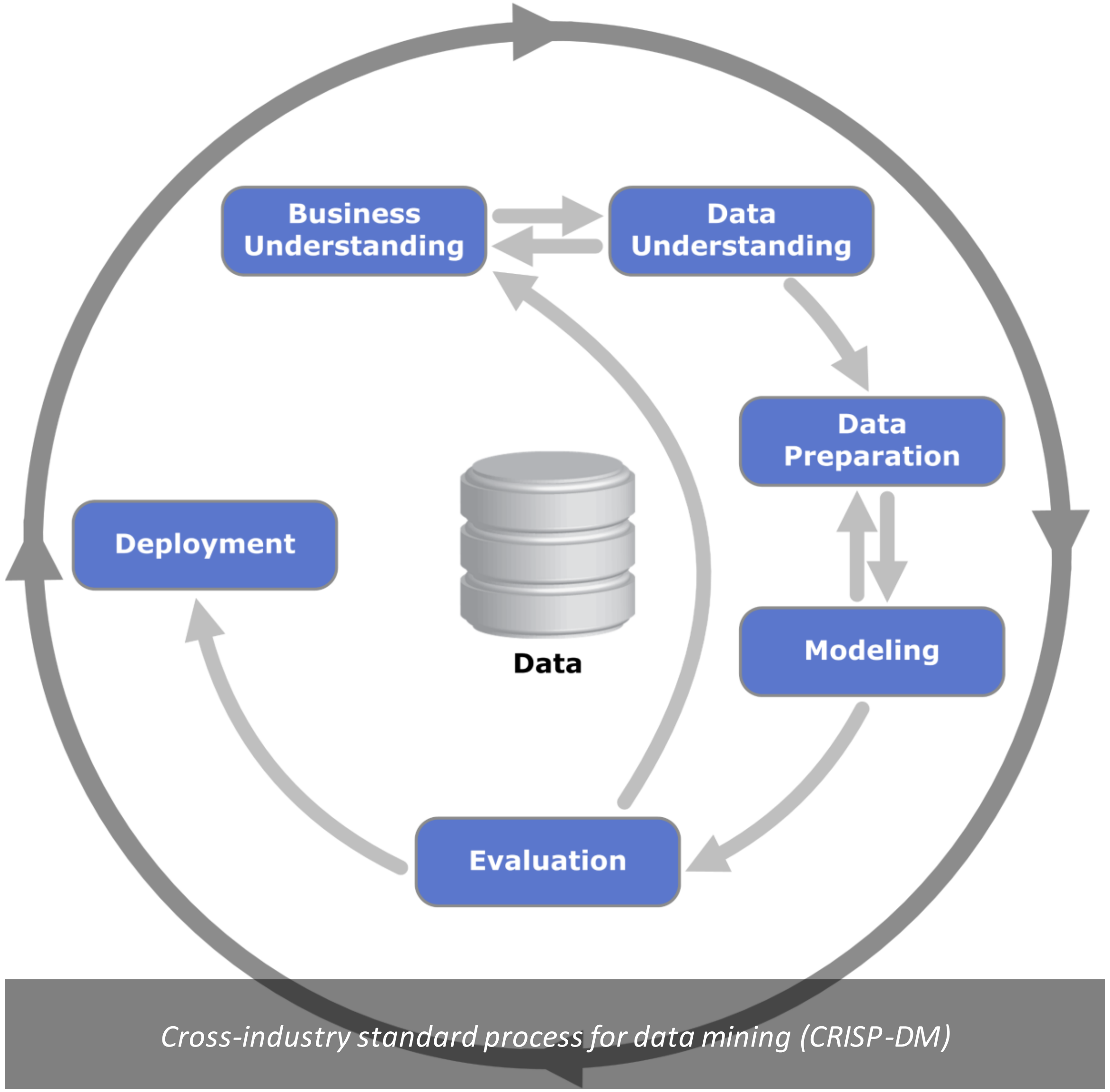
Bahasa Pemrograman

Data Preprocessing

Machine Learning dan Komputasi

Visualisasi Data

Pipeline



Cross-industry standard process for data mining (CRISP-DM)

Business Understanding

Latar belakang masalah

Penjualan rumah di King County, USA (2014-2015) dipengaruhi beberapa variabel

Tidak adanya acuan dalam menentukan harga rumah

Tujuan

Prediksi harga rumah

Mendapatkan insight dari data terhadap harga rumah dan penjualan

Data Understanding (1)

Terdiri dari 21613 baris dan 21 kolom (Target variable : *price*)

Variable	Description
id	Identification
date	Date sold
price	Sale price
bedrooms	Number of bedrooms
bathrooms	Number of bathrooms
sqft_liv	Size of living area in square feet
sqft_lot	Size of the lot in square feet
floors	Number of floors
waterfront	'1' if the property has a waterfront, '0' if not.
view	An index from 0 to 4 of how good the view of the property was
condition	Condition of the house, ranked from 1 to 5
grade	Classification by construction quality which refers to the types of materials used and the quality of workmanship. Buildings of better quality (higher grade) cost more to build per unit of measure and command higher value. Additional information in: KingCounty
sqft_above	Square feet above ground
sqft_basmt	Square feet below ground
yr_built	Year built
yr_renov	Year renovated. '0' if never renovated
zipcode	5 digit zip code
lat	Latitude
long	Longitude
squft_liv15	Average size of interior housing living space for the closest 15 houses, in square feet
squft_lot15	Average size of land lots for the closest 15 houses, in square feet

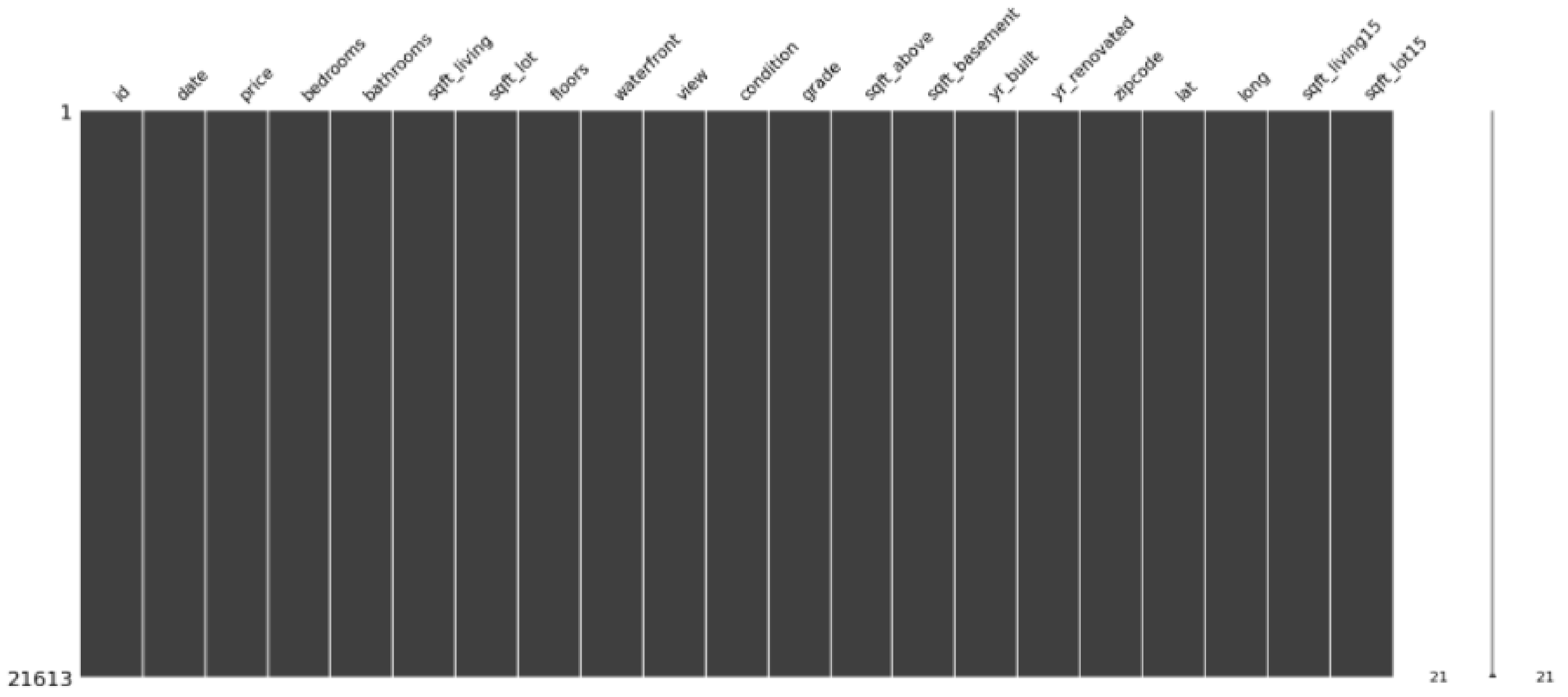
Data Understanding (2)

Melihat Sebagian Data

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
6420	5104531640	20150323T000000	585000.0	4	3.00	3400	5100	2.0	0	0	...	9	3400	0	2006	0	98038	47.3548	-122.002	3400	5672
10977	6065300570	20140624T000000	1250000.0	4	2.50	3220	15600	1.0	0	0	...	9	1680	1540	1973	0	98006	47.5697	-122.182	2990	15600
17259	8651720470	20140910T000000	506500.0	4	2.50	1890	7200	1.0	0	0	...	7	1500	390	1978	0	98034	47.7278	-122.218	2070	7200
21116	6824100029	20141031T000000	474950.0	3	3.00	1530	1568	3.0	0	0	...	8	1530	0	2012	0	98117	47.6998	-122.367	1460	1224
12908	2028701075	20140716T000000	626000.0	3	1.00	1040	4240	1.0	0	0	...	7	860	180	1924	0	98117	47.6768	-122.367	1170	4240
129	7853210060	20150406T000000	430000.0	4	2.50	2070	4310	2.0	0	0	...	7	2070	0	2004	0	98065	47.5319	-121.850	1970	3748
16959	7427800080	20150408T000000	626000.0	3	2.25	1810	5107	2.0	0	0	...	8	1810	0	1989	0	98033	47.6882	-122.171	1760	5454
9762	7229210060	20141211T000000	299950.0	3	1.75	1980	11274	1.0	0	0	...	7	1480	500	1968	0	98058	47.4474	-122.167	1520	8010
5594	3764650050	20140730T000000	463000.0	3	2.50	2010	4195	2.0	0	0	...	8	2010	0	1998	0	98034	47.7320	-122.197	2010	5779
293	6073240060	20141002T000000	580000.0	4	3.00	3280	11060	2.0	0	0	...	8	2270	1010	1986	0	98056	47.5399	-122.181	2320	11004

Data Understanding (3)

Melihat data yang hilang (null value)



Data Understanding (4)

Membagi kolom menjadi kolom numerik dan kategorik

```
numeric = ['price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors','sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated','lat',
           'long', 'sqft_living15', 'sqft_lot15']
```

```
categoric = ["grade","view","condition","zipcode","waterfront"]
```

Data Preparation

Transformasi Data

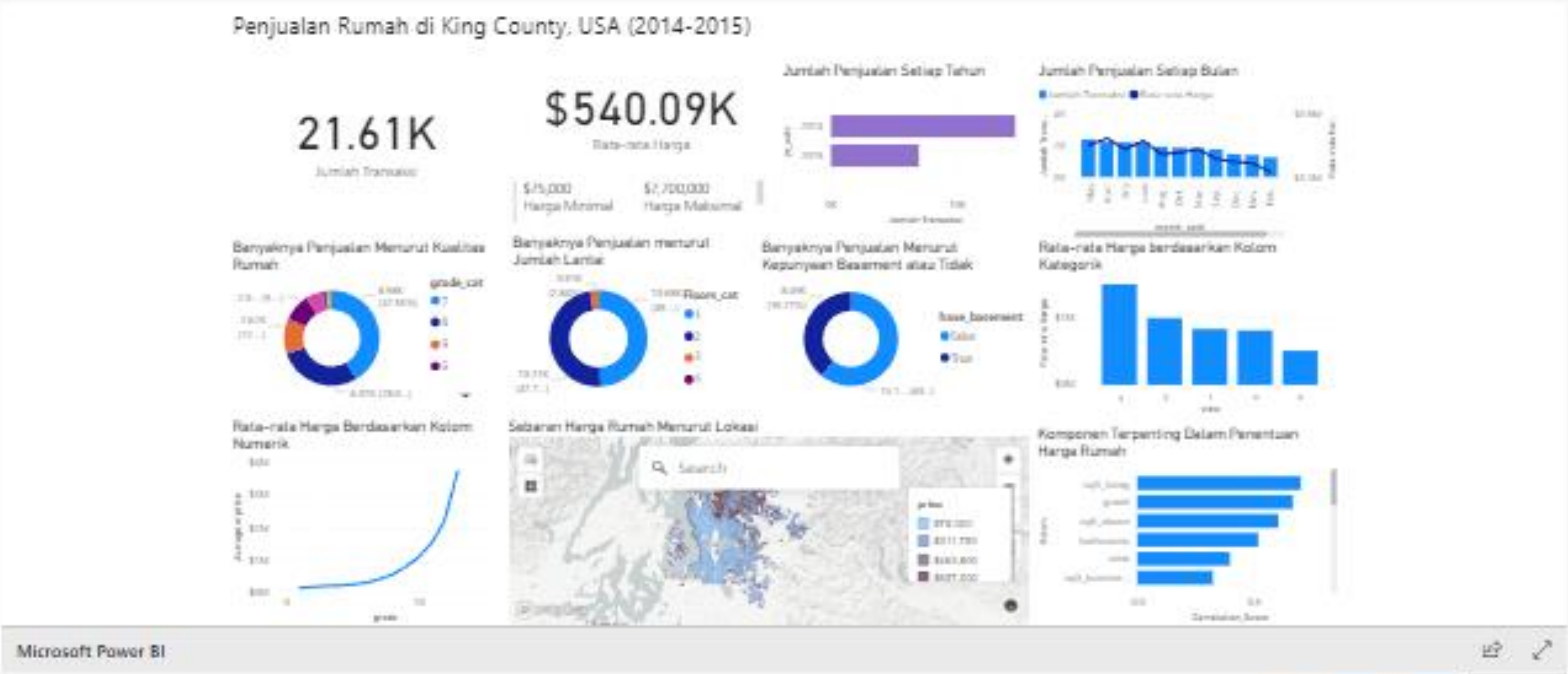
```
#Membulatkan bilangan pada variabel floors dan bathrooms
df["bathrooms"] = np.round(df.bathrooms)
df["floors"] = np.round(df.floors)
```

```
#Menambahkan kolom is_renovated
is_renovated = []
for x in df["yr_renovated"] :
    if x == 0:
        x=0
    else :
        x=1
    is_renovated.append(x)
df["is_renovated"] = np.array(is_renovated)
```

```
#Menambahkan kolom have_basement
have_basement = []
for x in df["sqft_basement"] :
    if x == 0:
        x=0
    else :
        x=1
    have_basement.append(x)
df["have_basement"] = np.array(have_basement)
```

```
#Menambahkan kolom building_age, yr_sold, month_sold
from datetime import datetime
import calendar

df['date'] = df['date'].str.split('T').str[0]

months = []
years = []
for x in df.date :
    datetime_object = datetime.strptime(x, '%Y%m%d')
    month = datetime_object.month
    month = calendar.month_name[month]
    year = datetime_object.year
    months.append(month)
    years.append(year)

df["yr_sold"] = np.array(years)
df["month_sold"] = np.array(months)
df["building_age"] = df["yr_sold"] - df["yr_built"]
```

Exploratory Data Analysis



Source: https://app.powerbi.com/view?

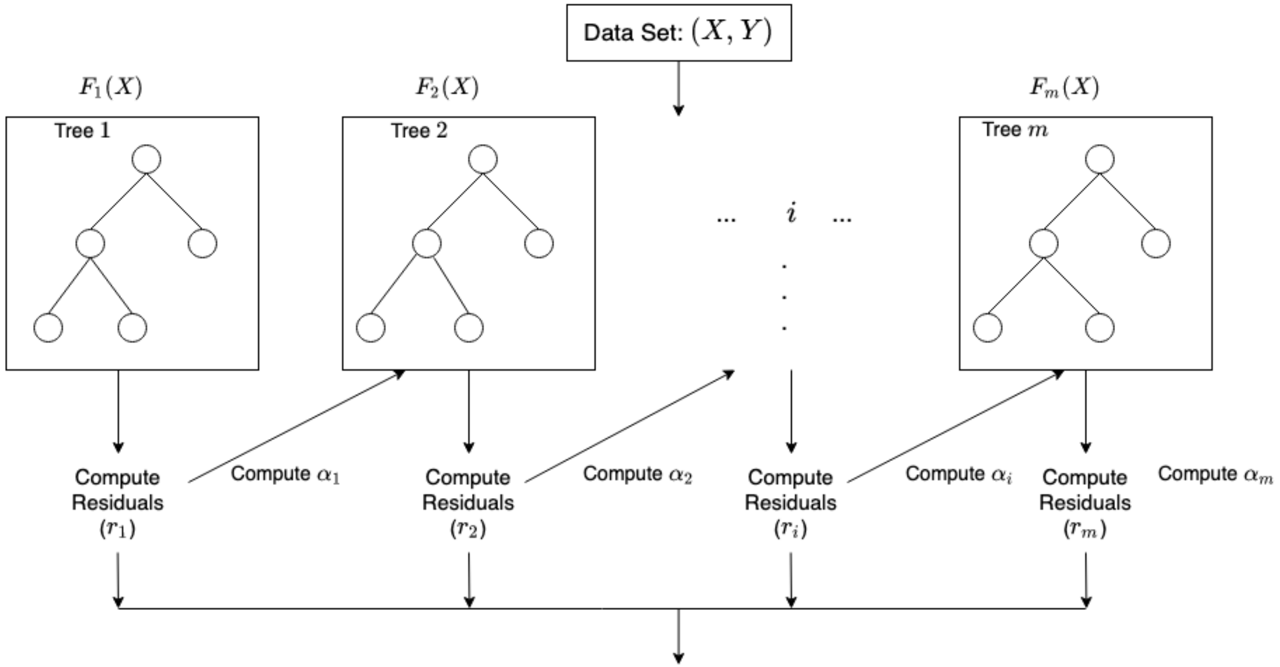
Modelling (XGBoost Regressor) [1]

Feature Selection (14 Variabel)

- "sqft_living15"
- "sqft_living"
- "sqft_above"
- "sqft_basement"
- "bathrooms"
- "bedrooms"
- "floors“
- "grade"
- "view"
- "have_basement"
- "waterfront"
- "is_renovated"
- "lat"
- "long"

Modelling (XGBoost Regressor) [2]

XGBoost Regressor



$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where α_i, and r_i are the regularization parameters and residuals computed with the i^{th} tree respectfully, and h_i is a function that is trained to predict residuals, r_i using X for the i^{th} tree. To compute α_i we use the residuals computed, r_i and compute the following: $arg \min_{\alpha} = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

Modelling (XGBoost Regressor)[3]

Hyperparameter Tuning

```python
#Parameter Tuning
def hyperParameterTuning(X_train, y_train):
    param_tuning = {
        'tree_method': ['gpu_hist'],
        'learning_rate': [0.01, 0.1],
        'max_depth': [3, 5, 7, 10],
        'min_child_weight': [1, 3, 5],
        'subsample': [0.5, 0.7],
        'colsample_bytree': [0.5, 0.7],
        'n_estimators' : [100, 200, 500],
        'objective': ['reg:squarederror']
    }

    xgb_model = XGBRegressor()

    gsearch = GridSearchCV(estimator = xgb_model,
                           param_grid = param_tuning,
                           #scoring = 'neg_mean_absolute_error', #MAE
                           scoring = 'neg_mean_squared_error',  #MSE
                           cv = 5,
                           n_jobs = -1,
                           verbose = 1)

    gsearch.fit(X_train,y_train)

    return gsearch.best_params_
```

```
Fitting 5 folds for each of 288 candidates, totalling 1440 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:   22.2s
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done 446 tasks      | elapsed: 11.4min
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed: 21.5min
[Parallel(n_jobs=-1)]: Done 1246 tasks      | elapsed: 33.8min
[Parallel(n_jobs=-1)]: Done 1440 out of 1440 | elapsed: 43.5min finished
{'colsample_bytree': 0.5,
 'learning_rate': 0.01,
 'max_depth': 7,
 'min_child_weight': 5,
 'n_estimators': 500,
 'objective': 'reg:squarederror',
 'subsample': 0.5,
 'tree_method': 'gpu_hist'}
```

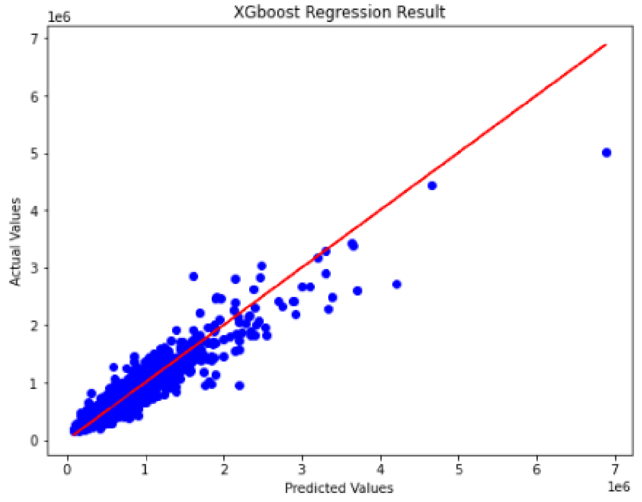
Modelling (XGBoost Regressor)[4]

Train Test Split

```
#Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

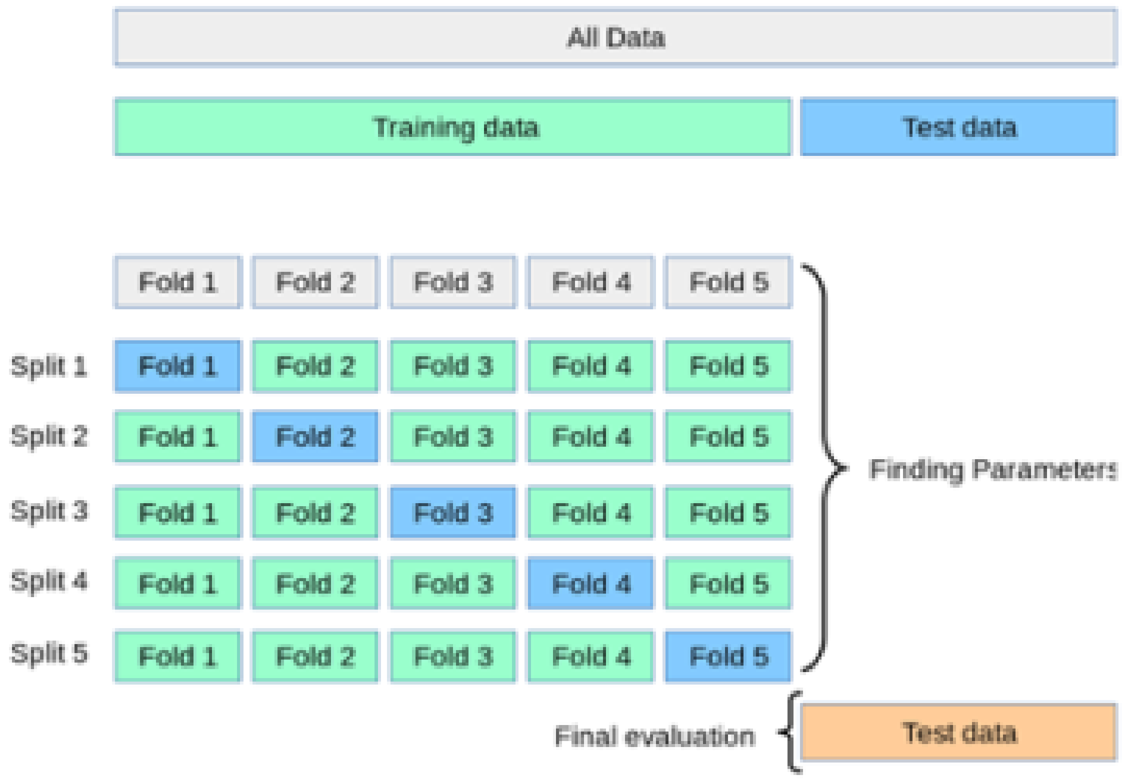
Training Model & Result

```
#Melatih Model
xgb_model = XGBRegressor(colsample_bytree = 0.5,
 learning_rate = 0.01,
 max_depth = 7,
 min_child_weight = 5,
 n_estimators = 500,
 objective = 'reg:squarederror',
 subsample= 0.5)
model = xgb_model.fit(X_train, y_train)

# Memprediksi nilai y dari X_test
y_predict = model.predict(X_test)
```



Evaluation

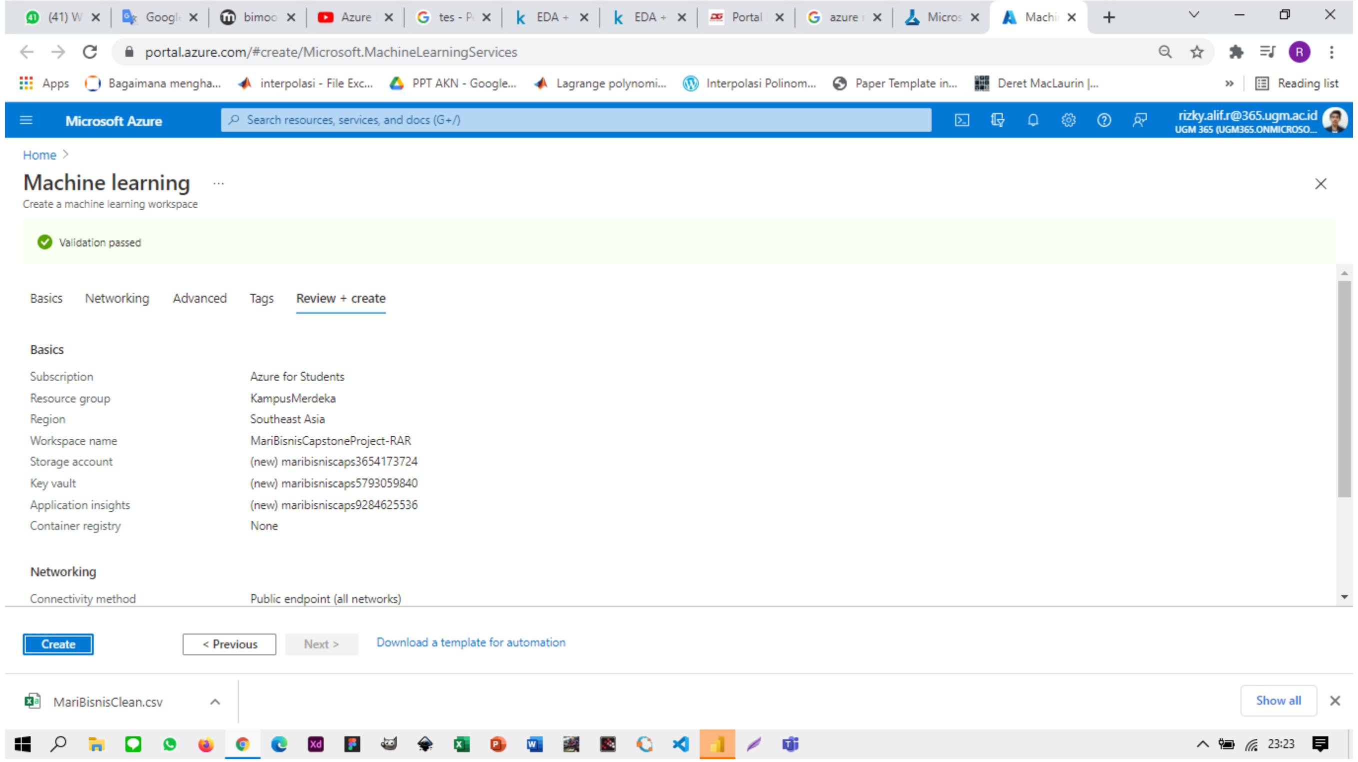
Cross-validation (fold = 5)




```
mean_RMSE :  129817.448992729696329
mean_MAE :  74566.775107059307629
mean_R2 :  0.874693459240766
```

Deployment (1)

Buat Machine Learning Resource



Deployment (2)

Save model dan registrasikan

```
#Menyimpan Model
from sklearn.externals import joblib
joblib.dump(value=model, filename="model.pkl")
```

<1 sec

```
from azureml.core import Workspace
ws = Workspace(subscription_id="760f001d-51de-4ea3-a6ab-57123ed2aba3",
               resource_group="KampusMerdeka",
               workspace_name="MariBisnisCapstoneProject-RAR")
```

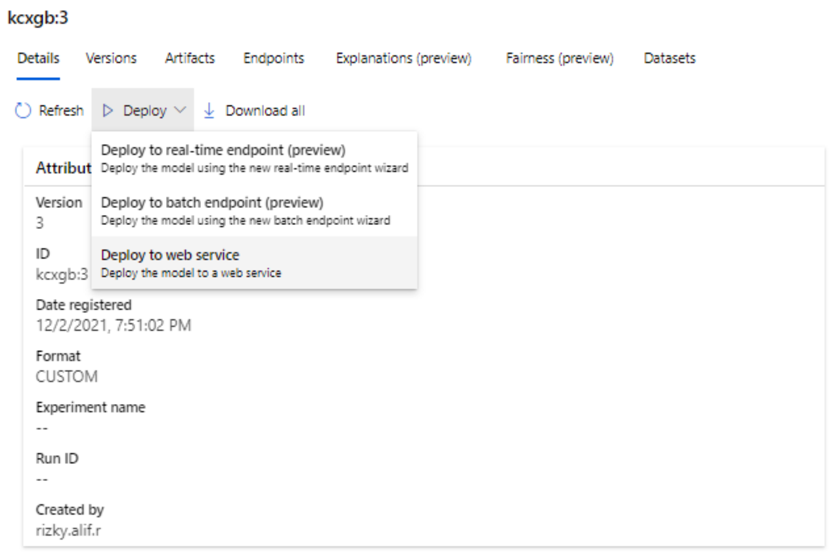
<1 sec

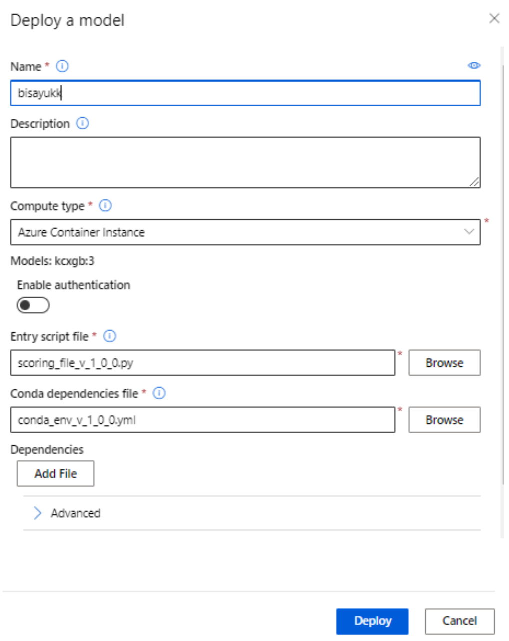
```
import urllib.request
from azureml.core.model import Model

# Register model
model = Model.register(ws, model_name="kcxgb", model_path="model.pkl")
```

2 sec

Deployment (3)

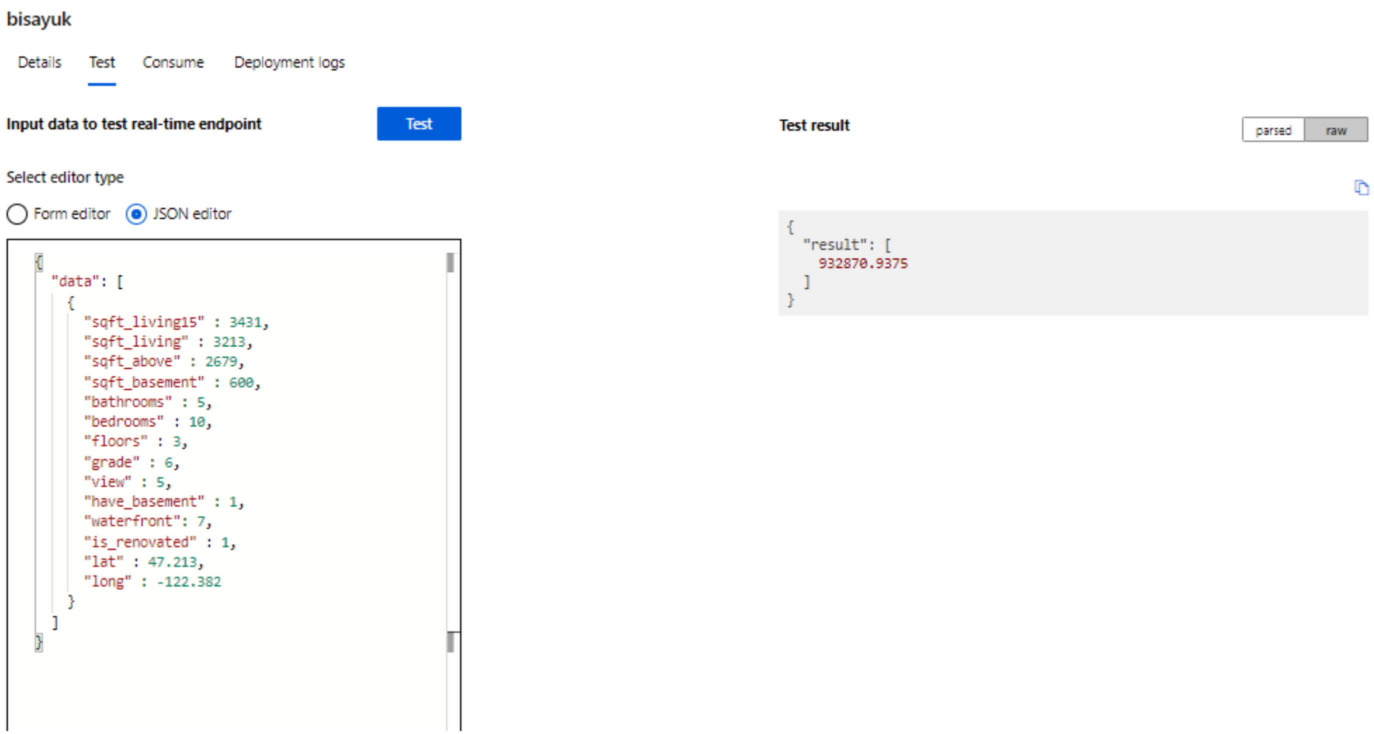
Deploy





Deployment (4)

Tes hasil *deployement*



Kesimpulan

Model berhasil dibuat dan di-*deploy*. Xgboost Regressor dengan parameter yang sudah di-*tuning* mampu memodelkan harga rumah dengan cukup baik, kecuali pada rumah-rumah yang terlampau mahal. Berikut hasil pemodelannya :

```
mean_RMSE :  129817.448992729696329
mean_MAE :  74566.775107059307629
mean_R2 :  0.874693459240766
```

Setelah melakukan EDA, 3 komponen terpenting yang memengaruhi harga rumah adalah luas bangunan, grade, dan luas bangunan di atas tanah

Terima kasih