Silicon valley real-estate project

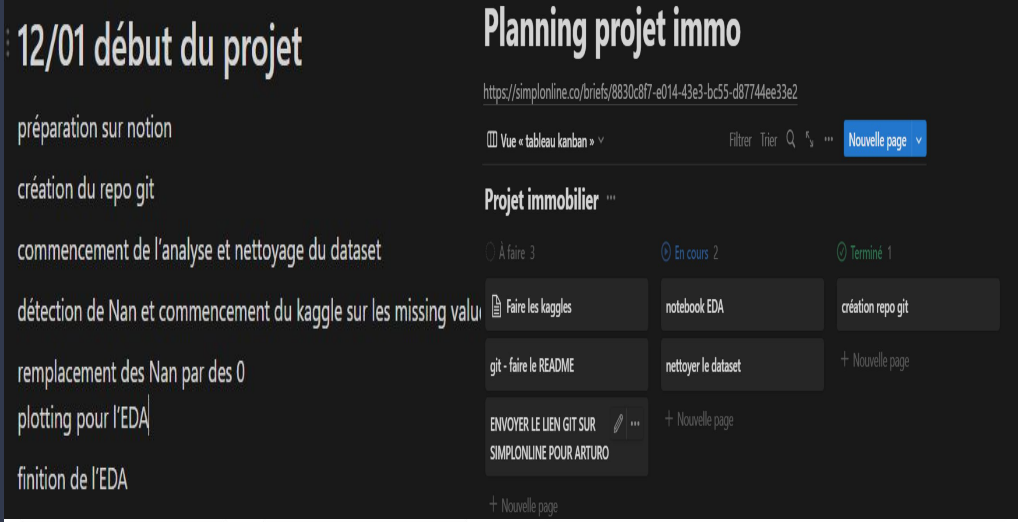
Maxime LU

Sommaire

Exploration:

Premier jour

Préparation
et
nettoyage



12/01 début du projet

préparation sur notion

création du repo git

commencement de l'analyse et nettoyage du dataset

détection de Nan et commencement du kaggle sur les missing valu

remplacement des Nan par des 0

plotting pour l'EDA

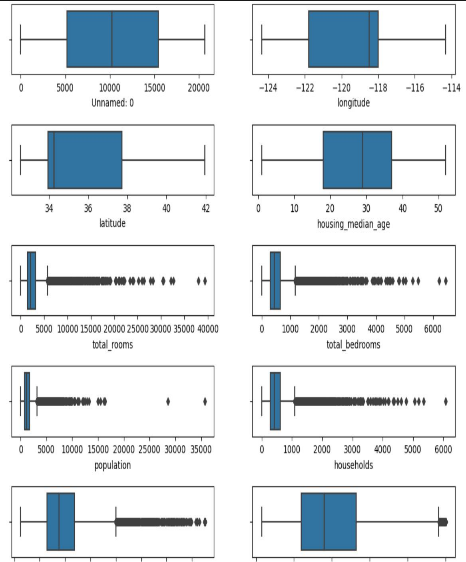
finition de l'EDA

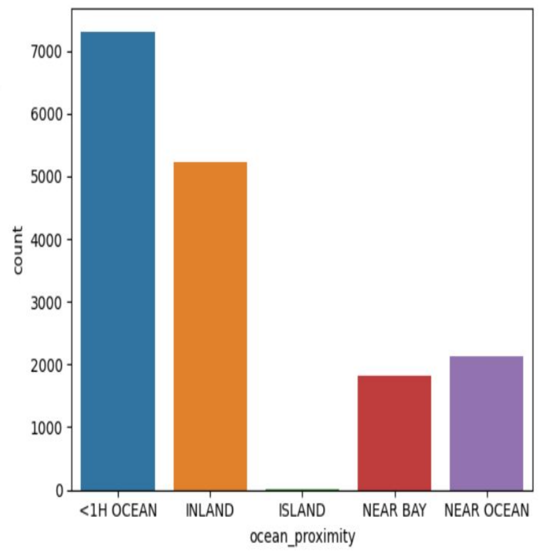
Planning projet immo

https://simplonline.co/briefs/8830c8f7-e014-43e3-bc55-d87744ee33e2

Projet immobilier

À faire 3	En cours 2	Terminé 1
Faire les kaggles	notebook EDA	création repo git
git - faire le README	nettoyer le dataset	+ Nouvelle page
ENVOYER LE LIEN GIT SUR SIMPLONLINE POUR ARTURO	+ Nouvelle page	
+ Nouvelle page		

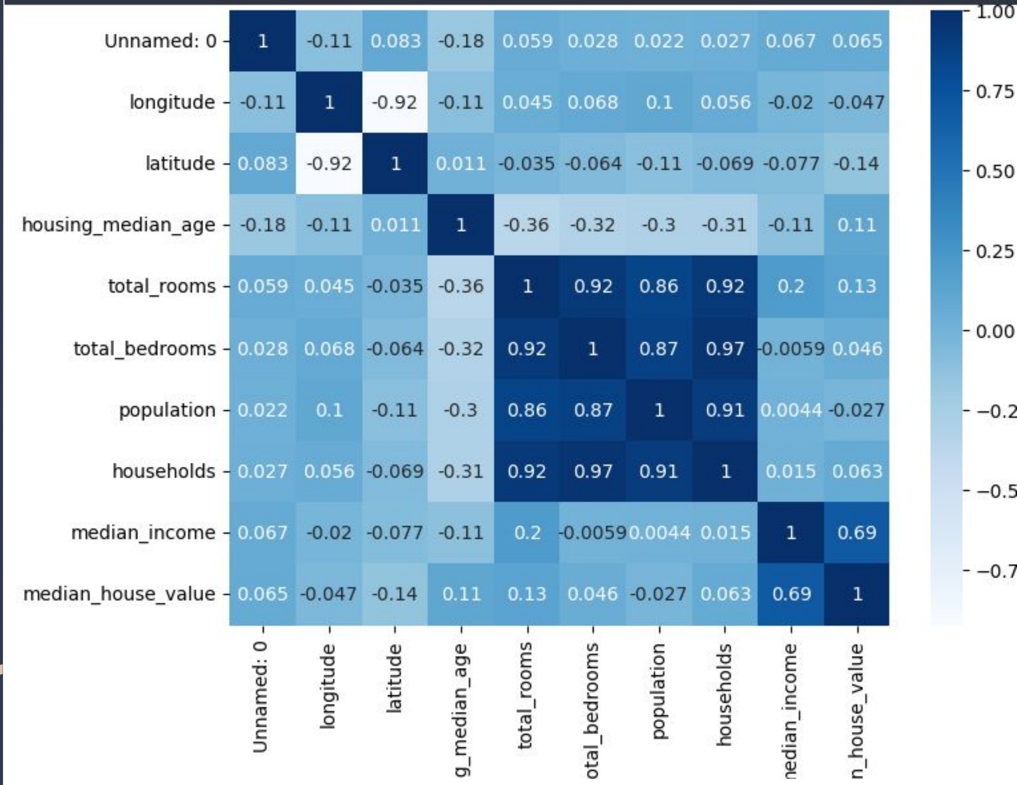




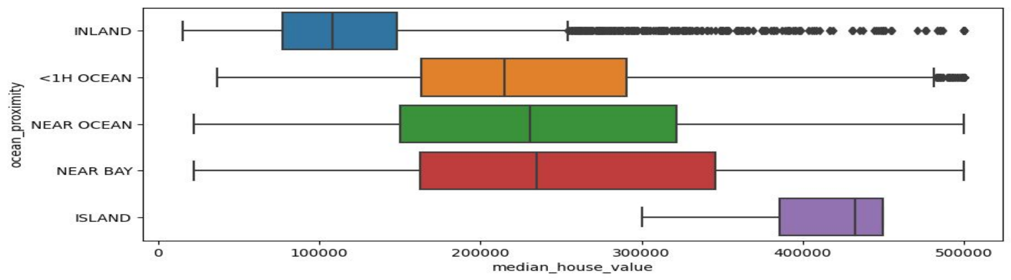
Exploration:

Deuxième jour

Notebook inférence



	Unnamed: 0	longitude	latitude	g_median_age	total_rooms	otal_bedrooms	population	households	nedian_income	n_house_value
Unnamed: 0	1	-0.11	0.083	-0.18	0.059	0.028	0.022	0.027	0.067	0.065
longitude	-0.11	1	-0.92	-0.11	0.045	0.068	0.1	0.056	-0.02	-0.047
latitude	0.083	-0.92	1	0.011	-0.035	-0.064	-0.11	-0.069	-0.077	-0.14
housing_median_age	-0.18	-0.11	0.011	1	-0.36	-0.32	-0.3	-0.31	-0.11	0.11
total_rooms	0.059	0.045	-0.035	-0.36	1	0.92	0.86	0.92	0.2	0.13
total_bedrooms	0.028	0.068	-0.064	-0.32	0.92	1	0.87	0.97	-0.0059	0.046
population	0.022	0.1	-0.11	-0.3	0.86	0.87	1	0.91	0.0044	-0.027
households	0.027	0.056	-0.069	-0.31	0.92	0.97	0.91	1	0.015	0.063
median_income	0.067	-0.02	-0.077	-0.11	0.2	-0.0059	0.0044	0.015	1	0.69
median_house_value	0.065	-0.047	-0.14	0.11	0.13	0.046	-0.027	0.063	0.69	1



Exploration:

Deuxième jour

Premiers models

13/01

push sur git

création data directory (j'ai mis un placeholder dedans pour le push vide)

création et personalisation de .gitignore

exportation de l'EDA

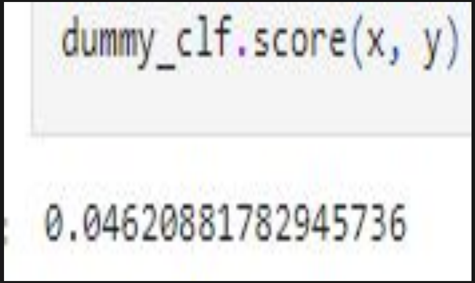
commencement de l'inférence

test de shapiro et levene pour l'anova

test de pearson

création du notebook model

création du premier model BM (score de 0,006)

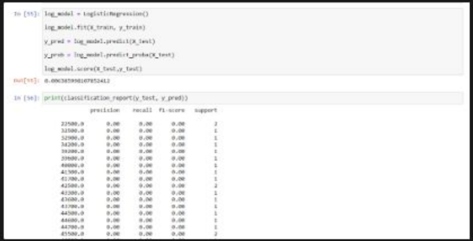


```
dummy_clf.score(x, y)

0.04620881782945736
```

test du score

ont peut voir le problème, je n'ai pas scale les donnée, c'est pour que sa mettait 10 a run.

je vais maintenant commencer le kaggle et recommencer le cycle a l'EDA

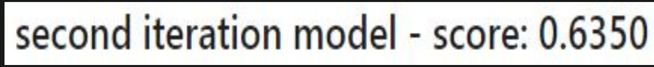


```
In [55]: log_model = LogisticRegression()
         log_model.fit(X_train, y_train)
         y_pred = log_model.predict(X_test)
         y_prob = log_model.predict_proba(X_test)
         log_model.score(X_test,y_test)
Out[55]: 0.006385998107852412

In [56]: print(classification_report(y_test, y_pred))
                   precision    recall  f1-score   support

         22500.0       0.00      0.00      0.00         2
         31500.0       0.00      0.00      0.00         1
         32900.0       0.00      0.00      0.00         1
         34200.0       0.00      0.00      0.00         1
         39200.0       0.00      0.00      0.00         1
         39600.0       0.00      0.00      0.00         1
         40000.0       0.00      0.00      0.00         1
         41300.0       0.00      0.00      0.00         1
         41700.0       0.00      0.00      0.00         1
         42500.0       0.00      0.00      0.00         2
         43300.0       0.00      0.00      0.00         1
         43600.0       0.00      0.00      0.00         1
         43700.0       0.00      0.00      0.00         1
         44500.0       0.00      0.00      0.00         1
         44600.0       0.00      0.00      0.00         1
         44700.0       0.00      0.00      0.00         1
         45500.0       0.00      0.00      0.00         2
```

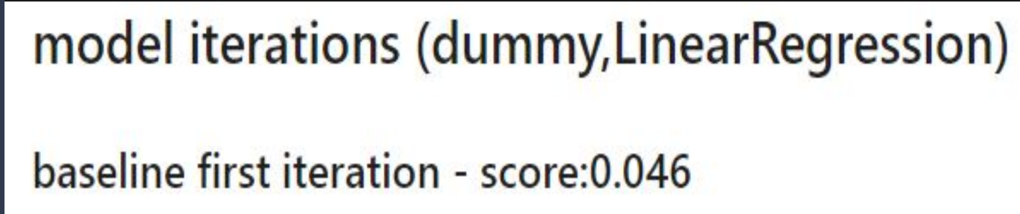
finition du tp model_evaluation

2eme itération de l'EDA

plot de la normalisation et analyse



second iteration model - score: 0.6350

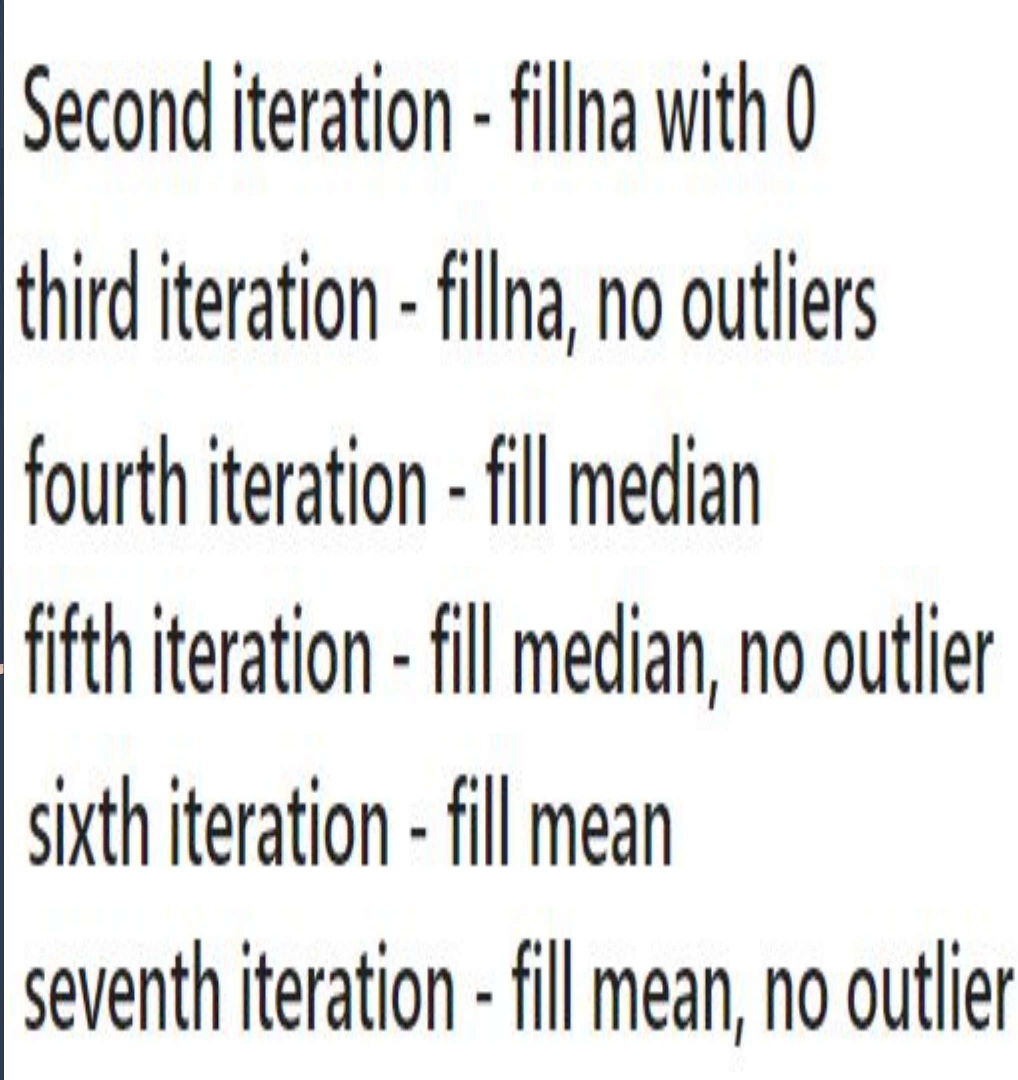


model iterations (dummy,LinearRegression)

baseline first iteration - score:0.046

Nettoyage:

des première iterations



Second iteration - fillna with 0

third iteration - fillna, no outliers

fourth iteration - fill median

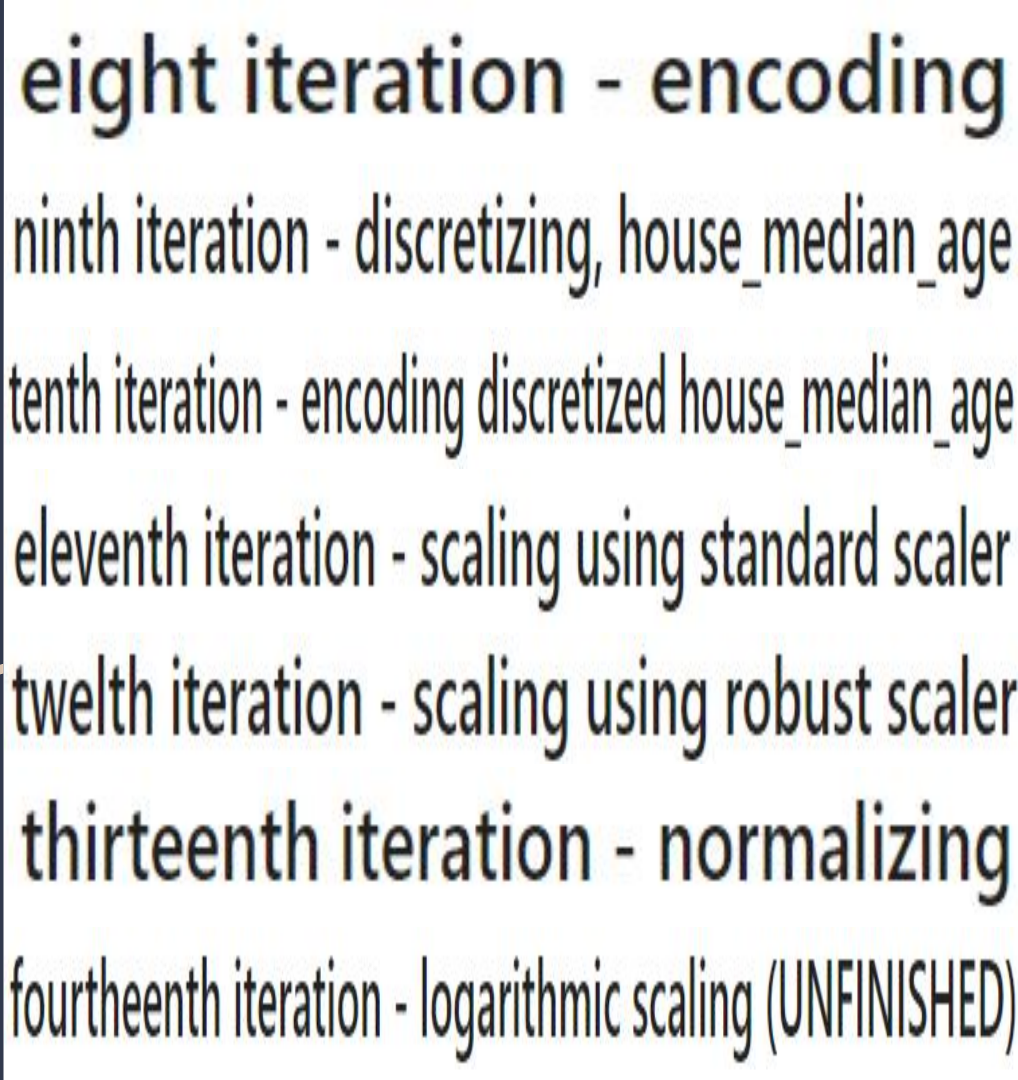
fifth iteration - fill median, no outlier

sixth iteration - fill mean

seventh iteration - fill mean, no outlier

Crash test:

création d’itération



eight iteration - encoding

ninth iteration - discretizing, house_median_age

tenth iteration - encoding discretized house_median_age

eleventh iteration - scaling using standard scaler

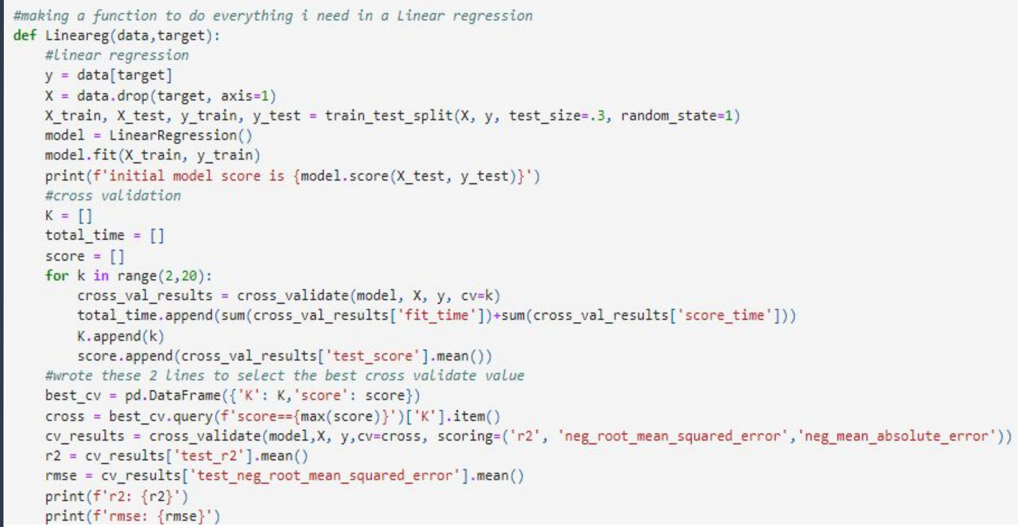
twelth iteration - scaling using robust scaler

thirteenth iteration - normalizing

fourtheenth iteration - logarithmic scaling (UNFINISHED)

Models:

Vérification des itérations



```python
#making a function to do everything i need in a Linear regression
def Lineareg(data,target):
    #linear regression
    y = data[target]
    X = data.drop(target, axis=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=1)
    model = LinearRegression()
    model.fit(X_train, y_train)
    print(f'initial model score is {model.score(X_test, y_test)}')
    #cross validation
    K = []
    total_time = []
    score = []
    for k in range(2,20):
        cross_val_results = cross_validate(model, X, y, cv=k)
        total_time.append(sum(cross_val_results['fit_time'])+sum(cross_val_results['score_time']))
        K.append(k)
        score.append(cross_val_results['test_score'].mean())
    #wrote these 2 lines to select the best cross validate value
    best_cv = pd.DataFrame({'K': K,'score': score})
    cross = best_cv.query(f'score=={max(score)}')['K'].item()
    cv_results = cross_validate(model,X, y,cv=cross, scoring=('r2', 'neg_root_mean_squared_error','neg_mean_absolute_error'))
    r2 = cv_results['test_r2'].mean()
    rmse = cv_results['test_neg_root_mean_squared_error'].mean()
    print(f'r2: {r2}')
    print(f'rmse: {rmse}')
```

tenth iteration model - score: 0.6498 best*

KNN - wanted to try other models so i see what i can get

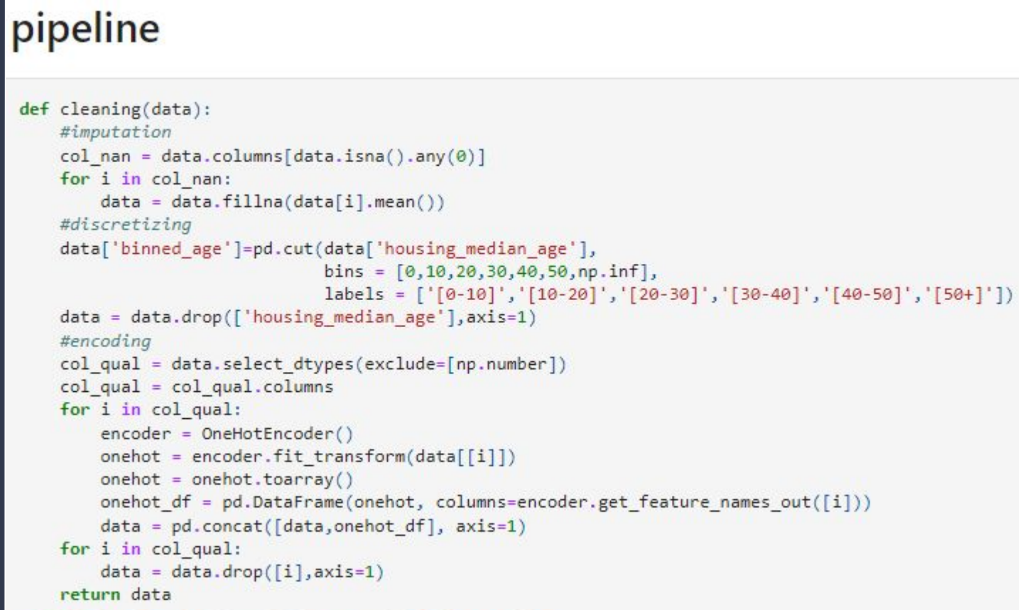
KNN function

DecisionTree - same as KNN

random forest - doesn't work at all

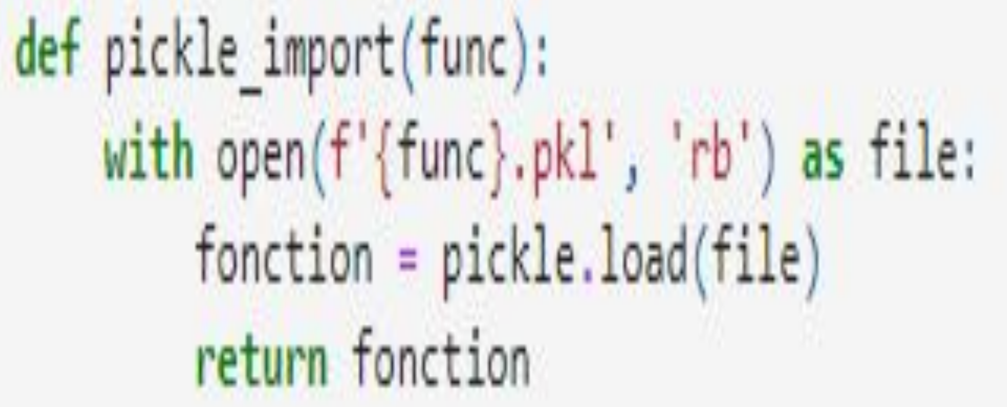
Makeshift pipeline:

notebook predict



pipeline

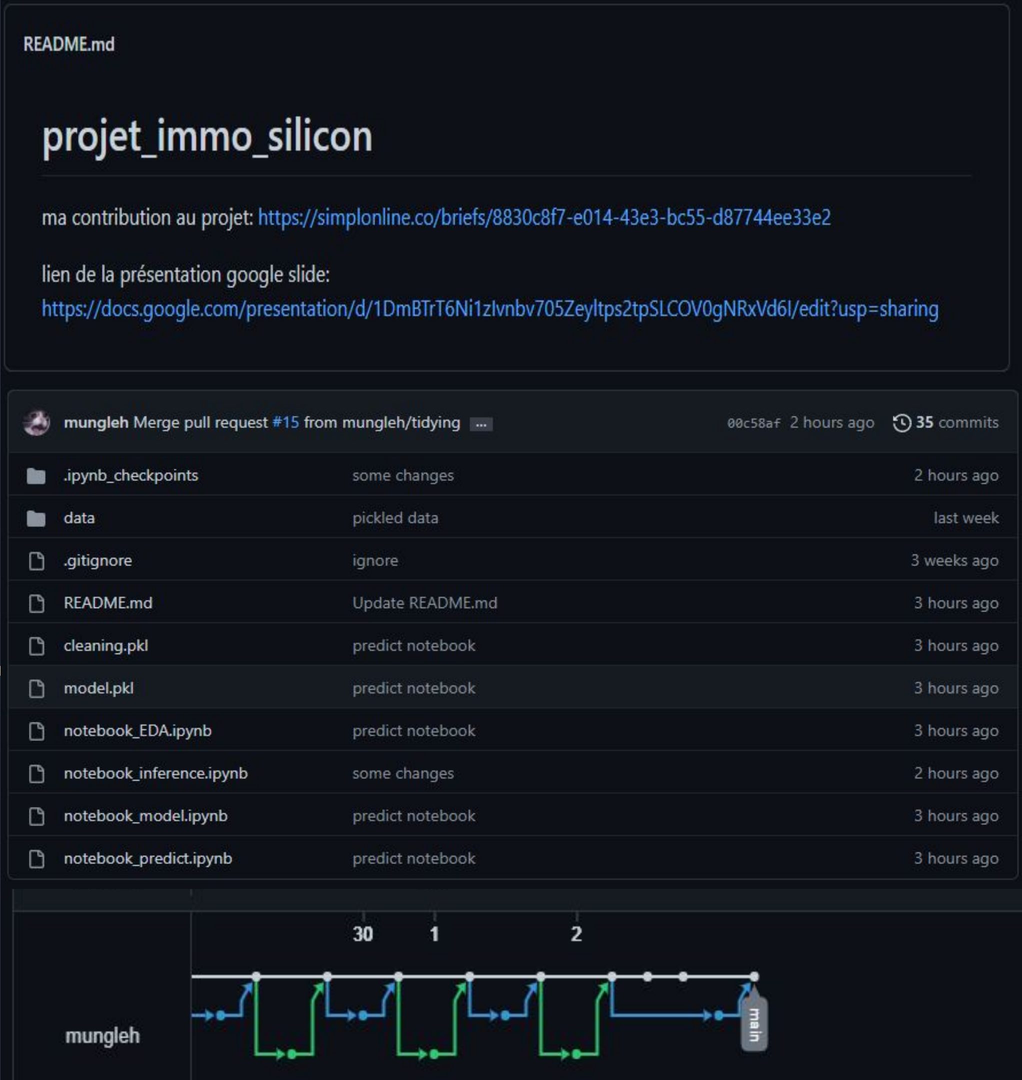
```
def cleaning(data):
    #imputation
    col_nan = data.columns[data.isna().any(0)]
    for i in col_nan:
        data = data.fillna(data[i].mean())
    #discretizing
    data['binned_age']=pd.cut(data['housing_median_age'],
                              bins = [0,10,20,30,40,50,np.inf],
                              labels = ['[0-10]','[10-20]','[20-30]','[30-40]','[40-50]','[50+]'])
    data = data.drop(['housing_median_age'],axis=1)
    #encoding
    col_qual = data.select_dtypes(exclude=[np.number])
    col_qual = col_qual.columns
    for i in col_qual:
        encoder = OneHotEncoder()
        onehot = encoder.fit_transform(data[[i]])
        onehot = onehot.toarray()
        onehot_df = pd.DataFrame(onehot, columns=encoder.get_feature_names_out([i]))
        data = pd.concat([data,onehot_df], axis=1)
    for i in col_qual:
        data = data.drop([i],axis=1)
    return data
```



```
def pickle_import(func):
    with open(f'{func}.pkl', 'rb') as file:
        fonction = pickle.load(file)
        return fonction
```

Git:

finalisation



README.md

projet_immo_silicon

ma contribution au projet: https://simplonline.co/briefs/8830c8f7-e014-43e3-bc55-d87744ee33e2

lien de la présentation google slide: https://docs.google.com/presentation/d/1DmBTrT6Ni1zlvnbv705Zeyltps2tpSLCOV0gNRxVd6l/edit?usp=sharing

mungleh Merge pull request #15 from mungleh/tidying … 00c58af 2 hours ago 35 commits

.ipynb_checkpoints	some changes	2 hours ago
data	pickled data	last week
.gitignore	ignore	3 weeks ago
README.md	Update README.md	3 hours ago
cleaning.pkl	predict notebook	3 hours ago
model.pkl	predict notebook	3 hours ago
notebook_EDA.ipynb	predict notebook	3 hours ago
notebook_inference.ipynb	some changes	2 hours ago
notebook_model.ipynb	predict notebook	3 hours ago
notebook_predict.ipynb	predict notebook	3 hours ago