1 Import Libraries

1.1 Data Processing Libraries

In [1]:

```
# data processing
import numpy as np
import pandas as pd
from scipy.stats.mstats import winsorize
from scipy import stats
```

1.2 Sklearn

In [2]:

```
# sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, KFold
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn import metrics
```

1.3 GLVQ & PSO manual libs

In [3]:

```
# model classes
from models.LearningVectorQuantization import LearningVectorQuantization as LVQ
from models.GeneralizedLearningVectorQuantization import GeneralizedLearningVectorQuantizat
from models.Utilization import Utilization
```

1.4 Utils

In [4]:

```
import random
import pickle

random_state = 22
random.seed(random_state)
```

1.5 Visualization

In [5]:

```
#import visualizing libraries
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

2 Load Data Training

In [6]:

```
wdbc_preprocessed = pickle.load(open('results/dataset_prep.pkl', 'rb'))
```

In [7]:

```
X_train = wdbc_preprocessed['X_train']
X_test = wdbc_preprocessed['X_test']
y_train = wdbc_preprocessed['y_train']
y_test = wdbc_preprocessed['y_test']
```

3 Load Predictors

Load only **kfold cross-validation**

In [8]:

```
kf_5 = pickle.load(open('results/predictors.pkl','rb'))['KFold']
n_splits = kf_5.get_n_splits()
```

4 Tuning Parameters GLVQ

- W: number of codebook in each class
- alpha: learning rate
- max_epoch: maximum iterations for learning stage
- min_error: minimum error allowed

4.1 Proses Tuning Parameter dengan Grid Search

In [9]:

```
codebooks = [1,2,3,4,5]
alphas = [round(i, 2) for i in np.arange(0.1, 1, 0.1)]
max_epochs = [100]
min_errors = [0.000001]
cross_val_results = list()
for codebook in codebooks:
    for alpha in alphas:
        for max_epoch in max_epochs:
            for min_error in min_errors:
                accuracy_score_list_per_combination = list()
                combination_name = "Codebook--"+str(codebook)+"_Alpha--"+str(alpha)+"_MaxEp
                accuracy_score_list_per_combination.append(combination_name)
                sum_acc = 0
                for train_index, validation_index in kf_5.split(X=X_train, y=y_train):
                    glvq = GLVQ(alpha=alpha, max_epoch=max_epoch, min_error=min_error, n_co
                    glvq.fit(X_train[train_index], y_train[train_index])
                    y_pred_val_glvq = glvq.predict(X_train[validation_index])
                    acc = metrics.accuracy_score(y_train[validation_index], y_pred_val_glvq
                    sum_acc += acc
                    accuracy_score_list_per_combination.append(acc)
                mean_accuracy_cross_validation = sum_acc/n_splits
                accuracy_score_list_per_combination.append(mean_accuracy_cross_validation)
                print(combination_name, mean_accuracy_cross_validation)
                cross_val_results.append(accuracy_score_list_per_combination)
```

```
Codebook--1_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.9340659340659341
Codebook--1_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--1_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--1_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--1_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--1_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--1_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--1_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--1_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--2_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--2_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9340659340659341
Codebook--2_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9406593406593406
Codebook--2_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.9362637362637362
Codebook--2_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--3_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.9538461538461538
Codebook--3_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--3_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--4_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.9516483516483516
Codebook--4_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--4_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--4_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9494505494505494
Codebook--4_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9472527472527472
```

```
Codebook--4_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--4_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--4_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9494505494505494
Codebook--4_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--5_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.956043956043956
Codebook--5_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.9494505494505494
Codebook--5_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9494505494505494
Codebook--5_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--5_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--5_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--5_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--5_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--5_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.945054945054945
```

In [10]:

```python
separator_parameter = "_"
separator_value = "--"
columns_name = ['combination_name'] + ["Fold-"+str(i+1) for i in range(n_splits)] + ['mean_

# rangkum hasil tuning parameter ke dalam bentuk Dataframe
glvq_tuning_parameters_results = pd.DataFrame(data=cross_val_results, columns=columns_name)
glvq_tuning_parameters_results['codebook'] = glvq_tuning_parameters_results.loc[:,'combinat
glvq_tuning_parameters_results['alpha'] = glvq_tuning_parameters_results.loc[:,'combination
glvq_tuning_parameters_results['max_epoch'] = glvq_tuning_parameters_results.loc[:,'combina
glvq_tuning_parameters_results['min_error'] = glvq_tuning_parameters_results.loc[:,'combina
```

In [11]:

```python
# cari hasil kombinasi parameter optimal dari hasil tuning parameter yang memberikan nilai
best_glvq_results = glvq_tuning_parameters_results[
    (glvq_tuning_parameters_results['mean_accuracy']==glvq_tuning_parameters_results['mean_
].reset_index(drop=True)
# random.randint(0, best_glvq_results.shape[0]-1)
p=0
optimal_codebook = int(best_glvq_results.loc[p,'codebook'])
optimal_alpha = float(best_glvq_results.loc[p,'alpha'])
optimal_max_epoch = int(best_glvq_results.loc[p,'max_epoch'])
optimal_min_error = float(best_glvq_results.loc[p,'min_error'])
```

In [12]:

```python
best_glvq_results
```

Out[12]:

	combination_name	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	mean_accuracy	codeb
0	Codebook--5_Alpha--0.1_MaxEpoch--100_MinError-...	0.989011	0.945055	0.934066	0.945055	0.967033	0.956044	

4.2 Simpan Hasil Tuning Parameter GLVQ

In [13]:

```
# Simpan hasil tuning parameter GLVQ dalam format csv
glvq_tuning_parameters_results.to_excel('informations/glvq_tuning_results.xlsx')
glvq_tuning_parameters_results.to_csv('informations/glvq_tuning_results.csv', index=False)
```

4.3 Simpan Parameter Optimal

In [14]:

```
optimal_parameters_glvq = {
    'optimal_codebook': optimal_codebook,
    'optimal_alpha': optimal_alpha,
    'optimal_max_epoch': optimal_max_epoch,
    'optimal_min_error': optimal_min_error,
}
pickle.dump(optimal_parameters_glvq, open('informations/optimal_parameters_glvq.pkl', 'wb')
```

5 Analisis Pengaruh Parameter

5.1 Analisis Pengaruh Parameter Alpha

5.1.1 Plotting Accuracy Alpha

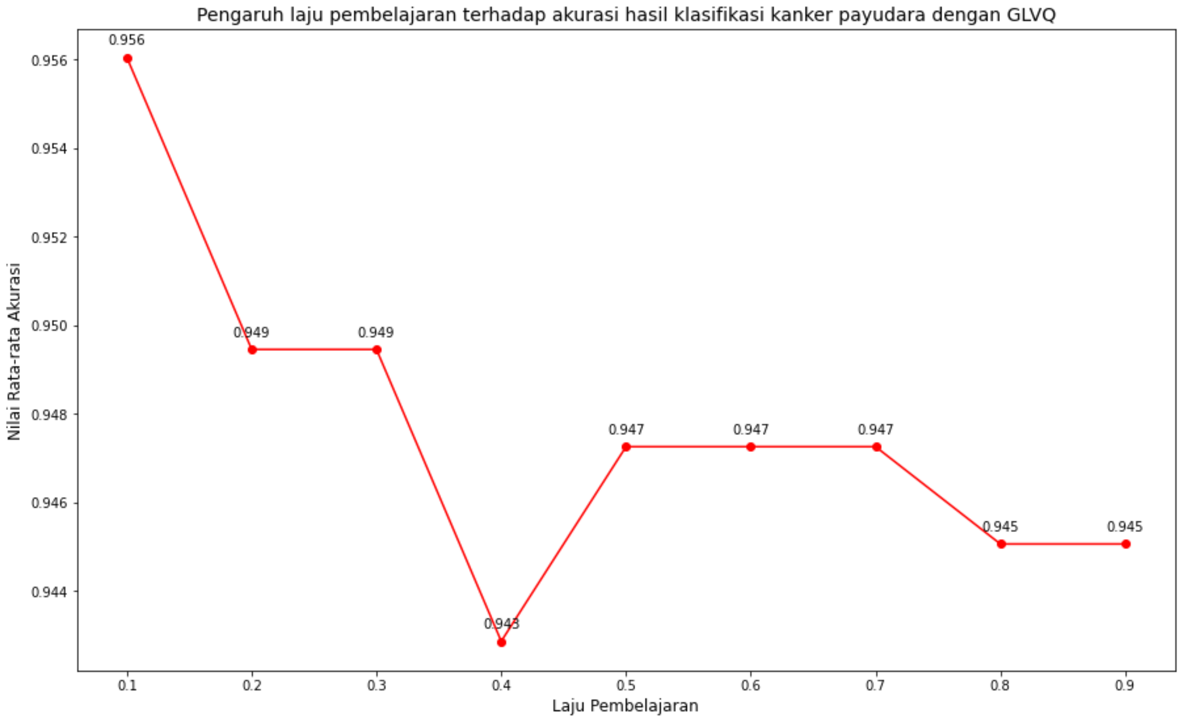
In [15]:

```
plt.figure(figsize=(15,9))
df_alpha = glvq_tuning_parameters_results[
    (glvq_tuning_parameters_results['codebook'] == optimal_codebook)
][['alpha','mean_accuracy']]
alphas = df_alpha['alpha']
mean_accuracy_alphas = df_alpha['mean_accuracy']


plt.plot(df_alpha['alpha'], df_alpha['mean_accuracy'], 'r-o')
plt.xticks(ticks=alphas)
plt.ylabel('Nilai Rata-rata Akurasi', size=12)
plt.xlabel('Laju Pembelajaran', size=12)
plt.title('Pengaruh laju pembelajaran terhadap akurasi hasil klasifikasi kanker payudara de

#zip alpha and mean accuracy for annotate graph
for alpha, mean_accuracy in zip(alphas, mean_accuracy_alphas):
    label = "{:.3f}".format(mean_accuracy)
    plt.annotate(
        label,
        (alpha,mean_accuracy),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )

plt.savefig('imageplot/alpha_acc.jpg')
plt.show()
```



5.1.2 Plotting Error Rate Alpha

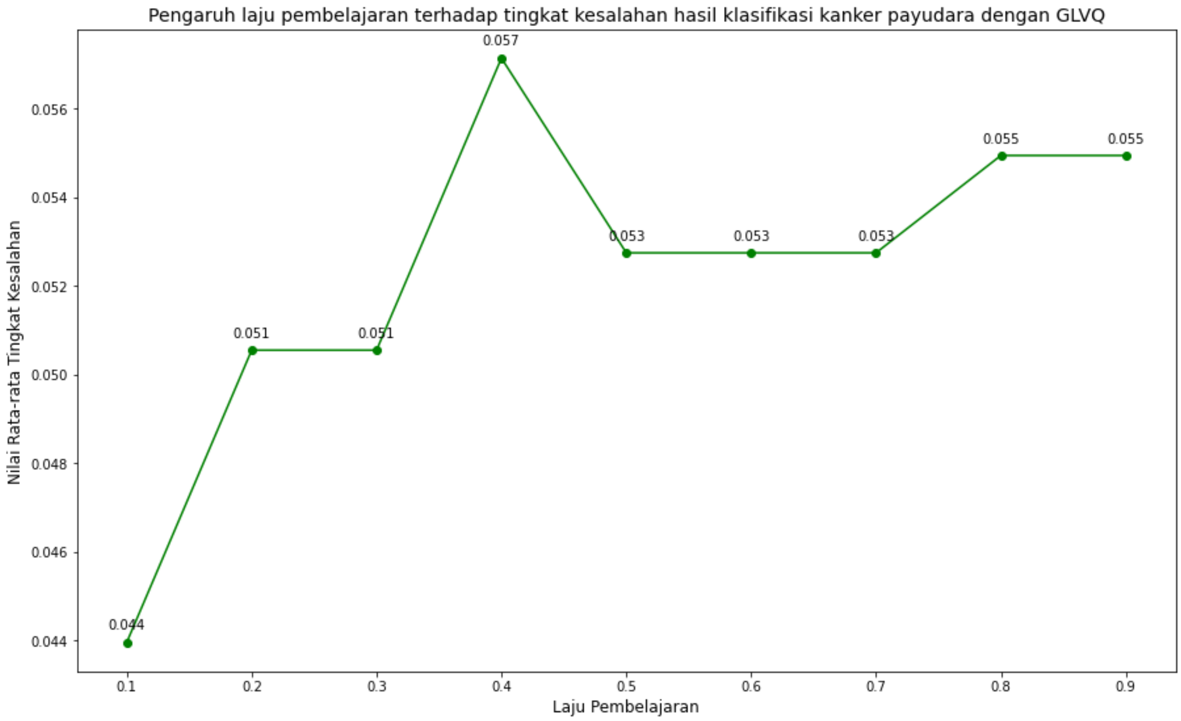
In [16]:

```
plt.figure(figsize=(15,9))
mean_errorrate_alpha = 1-df_alpha['mean_accuracy']


plt.plot(df_alpha['alpha'], 1-df_alpha['mean_accuracy'], 'g-o')
plt.xticks(ticks=alphas)
plt.ylabel('Nilai Rata-rata Tingkat Kesalahan', size=12)
plt.xlabel('Laju Pembelajaran', size=12)
plt.title('Pengaruh laju pembelajaran terhadap tingkat kesalahan hasil klasifikasi kanker p

#zip alpha and mean accuracy for annotate graph
for alpha, mean_error_rate in zip(alphas, mean_errorrate_alpha):
    label = "{:.3f}".format(mean_error_rate)
    plt.annotate(
        label,
        (alpha,mean_error_rate),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )

plt.savefig('imageplot/alpha_error.jpg')
plt.show()
```



5.2 Analisis Pengaruh Parameter Jumlah Prototype

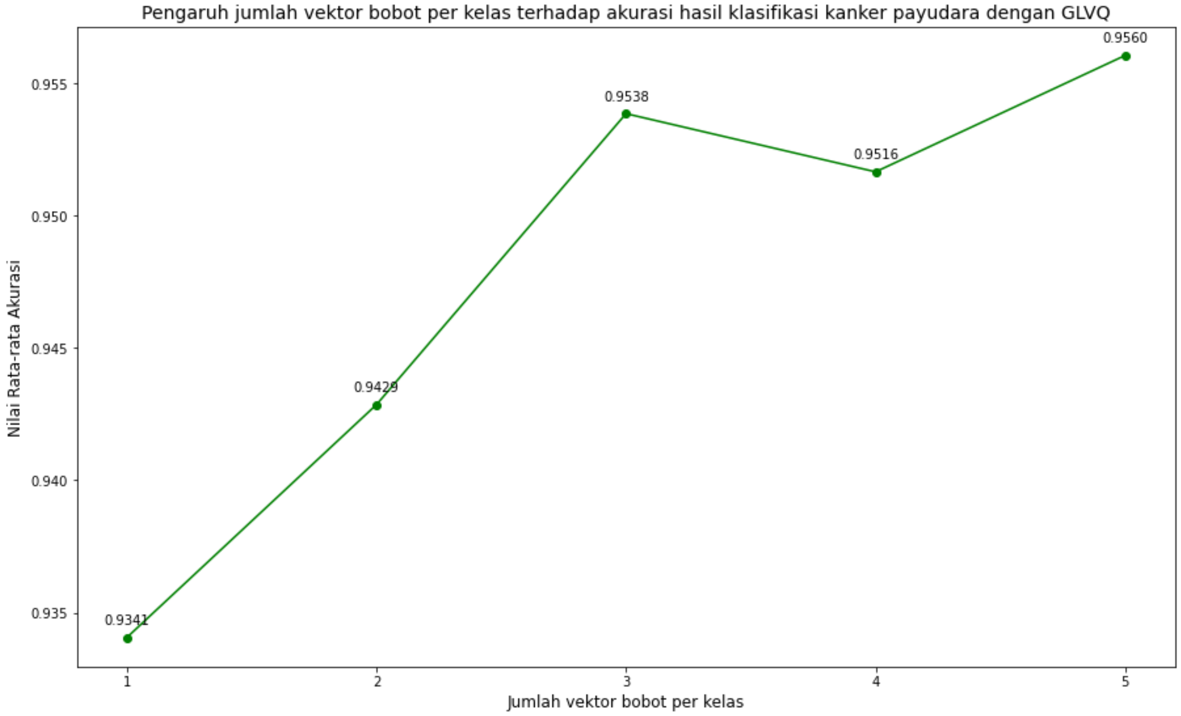
5.2.1 Plotting Accuracy Jumlah Prototype

In [17]:

```
plt.figure(figsize=(15,9))
df_codebook = glvq_tuning_parameters_results[
    (glvq_tuning_parameters_results['alpha'] == optimal_alpha)
][['codebook','mean_accuracy']]
codebooks = df_codebook['codebook']
mean_accuracy_codebooks = df_codebook['mean_accuracy']
plt.plot(df_codebook.set_index('codebook'), 'g-o')

plt.xticks(ticks=codebooks)
plt.ylabel('Nilai Rata-rata Akurasi', size=12)
plt.xlabel('Jumlah vektor bobot per kelas', size=12)
plt.title('Pengaruh jumlah vektor bobot per kelas terhadap akurasi hasil klasifikasi kanker
#zip codebook and mean accuracy for annotate graph
for codebook, mean_accuracy in zip(codebooks, mean_accuracy_codebooks):
    label = "{:.4f}".format(mean_accuracy)
    plt.annotate(
        label,
        (codebook,mean_accuracy),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )

plt.savefig('imageplot/codebook_acc.jpg')
plt.show()
```



5.2.2 Plotting Error Rate Jumlah Prototype

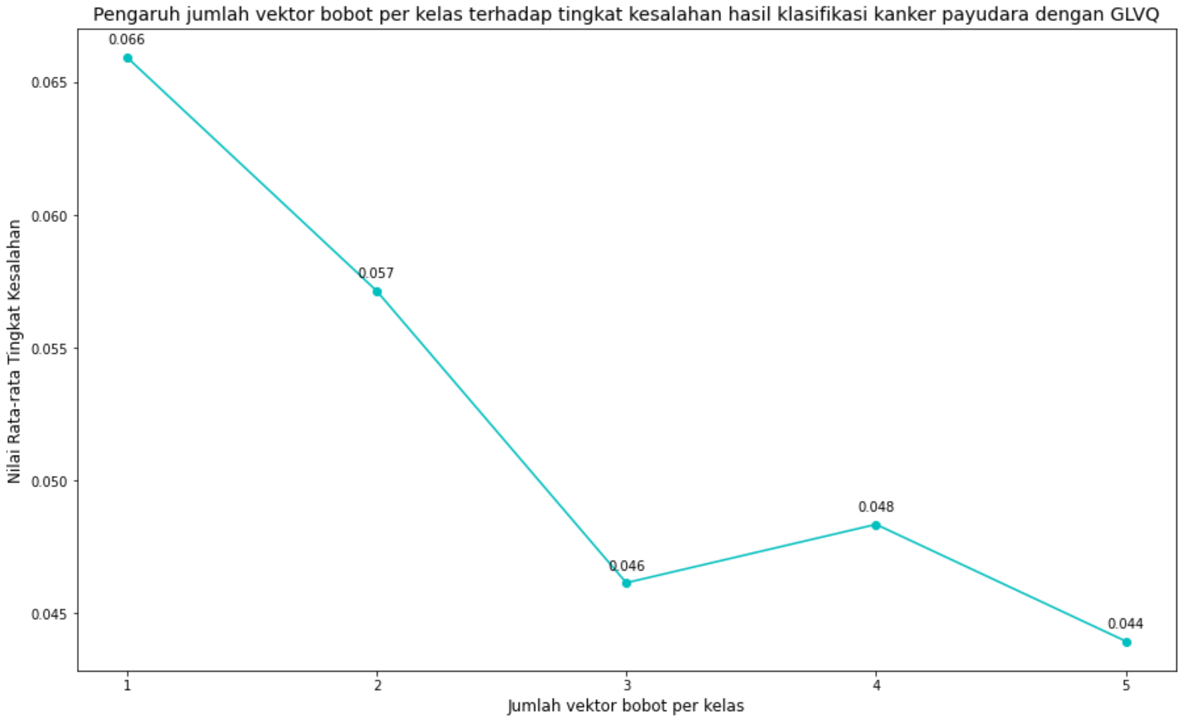
In [18]:

```
plt.figure(figsize=(15,9))
mean_errorrate_codebook = 1-df_codebook['mean_accuracy']


plt.plot(df_codebook['codebook'], 1-df_codebook['mean_accuracy'], 'c-o')
plt.xticks(ticks=codebooks)
plt.ylabel('Nilai Rata-rata Tingkat Kesalahan', size=12)
plt.xlabel('Jumlah vektor bobot per kelas', size=12)
plt.title('Pengaruh jumlah vektor bobot per kelas terhadap tingkat kesalahan hasil klasifik

#zip alpha and mean accuracy for annotate graph
for codebook, mean_error_rate in zip(codebooks, mean_errorrate_codebook):
    label = "{:.3f}".format(mean_error_rate)
    plt.annotate(
        label,
        (codebook,mean_error_rate),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )

plt.savefig('imageplot/codebook_error.jpg')
plt.show()
```



In [ ]:

```
```