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.ParticleSwarmOptimization import ParticleSwarmOptimization as PSO
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'))
hari_tuning_ke = "hari5"
```

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 GLVQ Tuning Parameter Result

In [8]:

```
optimal_parameters_glvq = pickle.load(open('informations/optimal_parameters_glvq.pkl','rb')
optimal_codebook = optimal_parameters_glvq['optimal_codebook']
```

In [9]:

```
optimal_codebook
```

Out[9]:

```
5
```

4 Load Predictors

Load only **kfold cross-validation**

In [10]:

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

5 Tuning Parameters PSO

- number of particles: 30
- maximum iteration: 100

- n_codebook --> using optimal codebook from tuning parameters GLVQ
- phi1 : [2.1, 2.2,..., 2.5]
- phi2 : [2.1, 2.2,..., 2.5]
- inertias: [0.5, 0.6, 0.7, 0.8, 0.9, 1.0] : Prebiana Journal

5.1 Proses Tuning Parameter PSO dengan GridSearch

In [11]:

```
list_phi1 = [2.5]
list_phi2 = [round(i,2) for i in np.arange(2.1, 2.6, 0.1)]
inertias = [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
maximum_iterations = [100]
n_particles = [30]
cross_val_results = list()
for phi1 in list_phi1:
    for phi2 in list_phi2:
        for inertia in inertias:
            for n_particle in n_particles:
                for max_iter in maximum_iterations:
                    accuracy_score_list_per_combination = list()
                    combination_name = "phi1-"+str(phi1)+"_phi2-"+str(phi2)+"_inertia-"+str
                    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):
                        pso = PSO(
                            n_particles=n_particle,
                            max_iter=max_iter,
                            phi1=phi1,
                            phi2=phi2,
                            n_codebooks=optimal_codebook,
                            inertia=inertia,
                            velocity_update="shi"
                        )
                        weight_pso_train = pso.fit(X_train[train_index], y_train[train_inde
                        y_pred_val_pso = pso.predict(X_train[validation_index])
                        acc = metrics.accuracy_score(y_train[validation_index], y_pred_val_
                        sum_acc+=acc
                        accuracy_score_list_per_combination.append(acc)
                    mean_accuracy_cross_validation = sum_acc/n_splits
                    print(combination_name, mean_accuracy_cross_validation)
                    accuracy_score_list_per_combination.append(mean_accuracy_cross_validati
                    cross_val_results.append(accuracy_score_list_per_combination)
```

```
phi1-2.5_phi2-2.1_inertia-0.5_nParticle-30_maxIter-100_Codebook-5 0.84835164
83516483
phi1-2.5_phi2-2.1_inertia-0.6_nParticle-30_maxIter-100_Codebook-5 0.85274725
27472527
phi1-2.5_phi2-2.1_inertia-0.7_nParticle-30_maxIter-100_Codebook-5 0.86373626
37362639
phi1-2.5_phi2-2.1_inertia-0.8_nParticle-30_maxIter-100_Codebook-5 0.90549450
54945055
phi1-2.5_phi2-2.1_inertia-0.9_nParticle-30_maxIter-100_Codebook-5 0.88131868
13186814
phi1-2.5_phi2-2.1_inertia-1.0_nParticle-30_maxIter-100_Codebook-5 0.87032967
03296705
phi1-2.5_phi2-2.2_inertia-0.5_nParticle-30_maxIter-100_Codebook-5 0.79780219
78021979
phi1-2.5_phi2-2.2_inertia-0.6_nParticle-30_maxIter-100_Codebook-5 0.83076923
07692307
phi1-2.5_phi2-2.2_inertia-0.7_nParticle-30_maxIter-100_Codebook-5 0.84175824
17582418
phi1-2.5_phi2-2.2_inertia-0.8_nParticle-30_maxIter-100_Codebook-5 0.82197802
19780219
phi1-2.5_phi2-2.2_inertia-0.9_nParticle-30_maxIter-100_Codebook-5 0.86813186
8131868
phi1-2.5_phi2-2.2_inertia-1.0_nParticle-30_maxIter-100_Codebook-5 0.82637362
```

```
63736263
phi1-2.5_phi2-2.3_inertia-0.5_nParticle-30_maxIter-100_Codebook-5 0.74725274
72527473
phi1-2.5_phi2-2.3_inertia-0.6_nParticle-30_maxIter-100_Codebook-5 0.79120879
12087912
phi1-2.5_phi2-2.3_inertia-0.7_nParticle-30_maxIter-100_Codebook-5 0.87252747
25274724
phi1-2.5_phi2-2.3_inertia-0.8_nParticle-30_maxIter-100_Codebook-5 0.86153846
15384617
phi1-2.5_phi2-2.3_inertia-0.9_nParticle-30_maxIter-100_Codebook-5 0.89230769
23076924
phi1-2.5_phi2-2.3_inertia-1.0_nParticle-30_maxIter-100_Codebook-5 0.85274725
27472527
phi1-2.5_phi2-2.4_inertia-0.5_nParticle-30_maxIter-100_Codebook-5 0.85934065
93406593
phi1-2.5_phi2-2.4_inertia-0.6_nParticle-30_maxIter-100_Codebook-5 0.84835164
83516483
phi1-2.5_phi2-2.4_inertia-0.7_nParticle-30_maxIter-100_Codebook-5 0.83076923
07692308
phi1-2.5_phi2-2.4_inertia-0.8_nParticle-30_maxIter-100_Codebook-5 0.82637362
63736263
phi1-2.5_phi2-2.4_inertia-0.9_nParticle-30_maxIter-100_Codebook-5 0.80659340
65934066
phi1-2.5_phi2-2.4_inertia-1.0_nParticle-30_maxIter-100_Codebook-5 0.78901098
90109891
phi1-2.5_phi2-2.5_inertia-0.5_nParticle-30_maxIter-100_Codebook-5 0.84395604
3956044
phi1-2.5_phi2-2.5_inertia-0.6_nParticle-30_maxIter-100_Codebook-5 0.85054945
05494505
phi1-2.5_phi2-2.5_inertia-0.7_nParticle-30_maxIter-100_Codebook-5 0.86373626
37362637
phi1-2.5_phi2-2.5_inertia-0.8_nParticle-30_maxIter-100_Codebook-5 0.84395604
3956044
phi1-2.5_phi2-2.5_inertia-0.9_nParticle-30_maxIter-100_Codebook-5 0.85714285
71428571
phi1-2.5_phi2-2.5_inertia-1.0_nParticle-30_maxIter-100_Codebook-5 0.84395604
3956044
```

In [12]:

```
columns_name = ['combination_name'] + ["Fold-"+str(i+1) for i in range(n_splits)] + ['mean_
pso_tuning_parameters_results = pd.DataFrame(data=cross_val_results, columns=columns_name)
pso_tuning_parameters_results['phi1'] = pso_tuning_parameters_results.loc[:,'combination_na
pso_tuning_parameters_results['phi2'] = pso_tuning_parameters_results.loc[:,'combination_na
pso_tuning_parameters_results['inertia'] = pso_tuning_parameters_results.loc[:,'combination
pso_tuning_parameters_results['n_particles'] = pso_tuning_parameters_results.loc[:,'combina
pso_tuning_parameters_results['max_iter'] = pso_tuning_parameters_results.loc[:,'combinatio
```

5.2 Simpan Hasil Tuning Parameter PSO

In [13]:

```
filename = "psoshi_tuning_parameters_results_"+hari_tuning_ke+".csv"
filename1 = "psoshi_tuning_parameters_results_"+hari_tuning_ke+".xlsx"
pso_tuning_parameters_results.to_csv('informations/'+filename, index=False)
pso_tuning_parameters_results.to_excel('informations/'+filename1)
```

5.3 Simpan Parameter Optimal PSO

5.3.1 Baca seluruh hasil training setiap hari

In [28]:

```
pso_shi_tuning_results = list()
for idxhari in range(1,6,1):
    file_name = "informations/psoshi_tuning_parameters_results_hari"+str(idxhari)+".csv"
    pso_shi_tuning_results.append(pd.read_csv(file_name))
```

5.3.2 Merge semua dataframe seluruh hasil training

In [29]:

```
pso_tuning_parameters_results = pd.concat(pso_shi_tuning_results, ignore_index=True)
```

5.3.3 Simpan Dataframe yang sudah dimerge

In [30]:

```
pso_tuning_parameters_results.to_csv('informations/psoshi_tuning_parameters_results.csv', i
pso_tuning_parameters_results.to_excel('informations/psoshi_tuning_parameters_results.xlsx'
```

5.3.4 Cari parameter optimal dari keseluruhan hasil training

In [31]:

```
best_pso_results = pso_tuning_parameters_results[
    (pso_tuning_parameters_results['mean_accuracy']==pso_tuning_parameters_results['mean_ac
].reset_index(drop=True)
optimal_phi1 = float(best_pso_results.loc[0,'phi1'])
optimal_phi2 = float(best_pso_results.loc[0,'phi2'])
optimal_inertia = float(best_pso_results.loc[0,'inertia'])
optimal_n_particles = int(best_pso_results.loc[0,'n_particles'])
optimal_max_iter = int(best_pso_results.loc[0,'max_iter'])
```

In [39]:

```
best_pso_results
```

Out[39]:

	combination_name	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	mean_accuracy	phi1	p
0	phi1-2.4_phi2-2.1_inertia-0.6_nParticle-30_max...	0.89011	0.923077	0.912088	0.923077	0.945055	0.918681	2.4	

5.3.4.1 Simpan nilai parameter optimal dari PSO dari keseluruhan hasil training

In [32]:

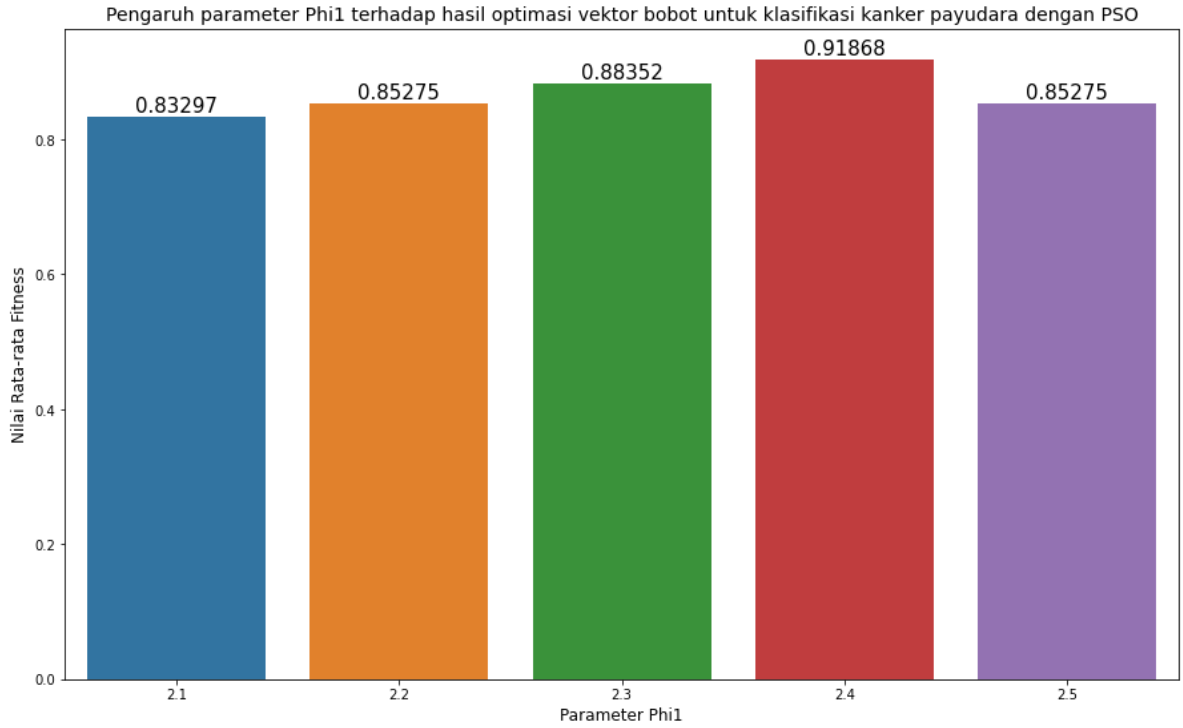
```
optimal_parameters_pso_shi = {
    'optimal_phi1': optimal_phi1,
    'optimal_phi2': optimal_phi2,
    'optimal_inertia': optimal_inertia,
    'optimal_n_particles': optimal_n_particles,
    'optimal_max_iter': optimal_max_iter,
}
pickle.dump(optimal_parameters_pso_shi, open('informations/optimal_parameters_pso_shi.pkl',
```

6 Analisis Pengaruh Parameter

6.1 Analisis Pengaruh Parameter Phi1

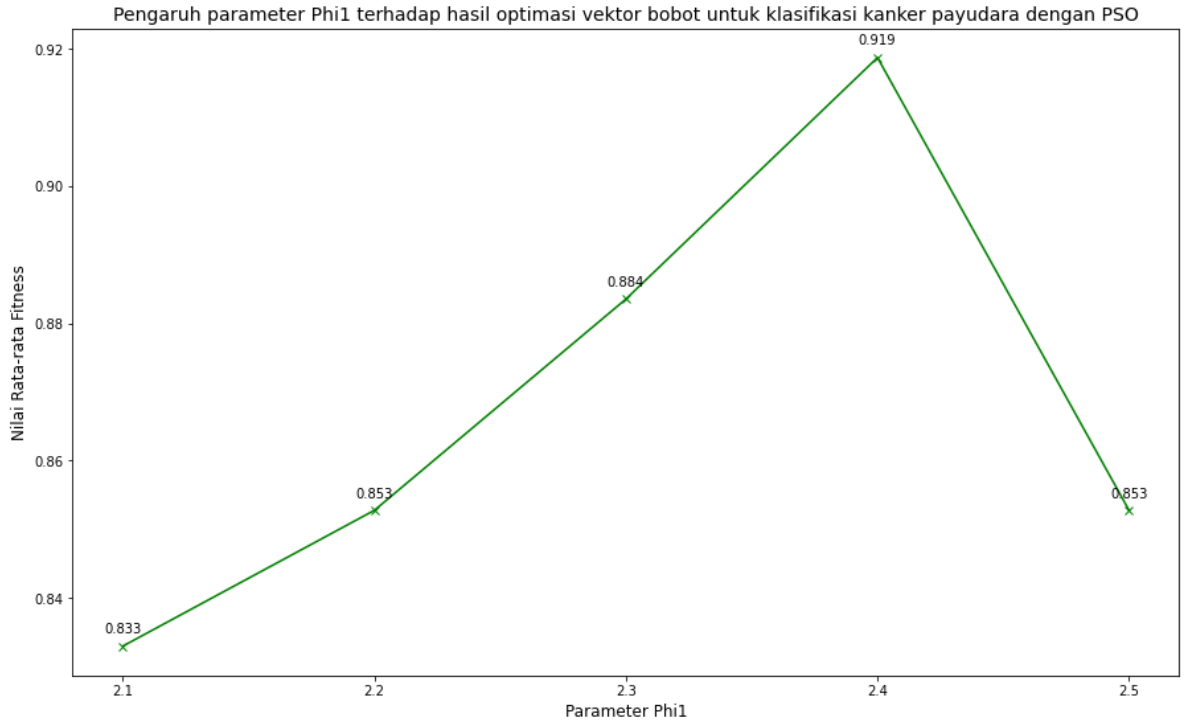
In [33]:

```
df_phi1 = pso_tuning_parameters_results[
    (pso_tuning_parameters_results['phi2'] == optimal_phi2) &
    (pso_tuning_parameters_results['inertia'] == optimal_inertia) &
    (pso_tuning_parameters_results['n_particles'] == optimal_n_particles)
][['phi1', 'mean_accuracy']]

# Defining the plot size
plt.figure(figsize=(15, 9))
plots = sns.barplot(x="phi1", y="mean_accuracy", data=df_phi1)
plt.ylabel('Nilai Rata-rata Fitness', size=12)
plt.xlabel('Parameter Phi1', size=12)
plt.title('Pengaruh parameter Phi1 terhadap hasil optimasi vektor bobot untuk klasifikasi k
for bar in plots.patches:
    # Using Matplotlib's annotate function and
    # passing the coordinates where the annotation shall be done
    # x-coordinate: bar.get_x() + bar.get_width() / 2
    # y-coordinate: bar.get_height()
    # free space to be left to make graph pleasing: (0, 8)
    # ha and va stand for the horizontal and vertical alignment
    plots.annotate(format(bar.get_height(), '.5f'),
                   (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                   ha='center',
                   va='center',
                   size=15,
                   xytext=(0, 8),
                   textcoords='offset points')
plt.savefig('imageplot/phi1.jpg')
plt.show()
```



In [34]:

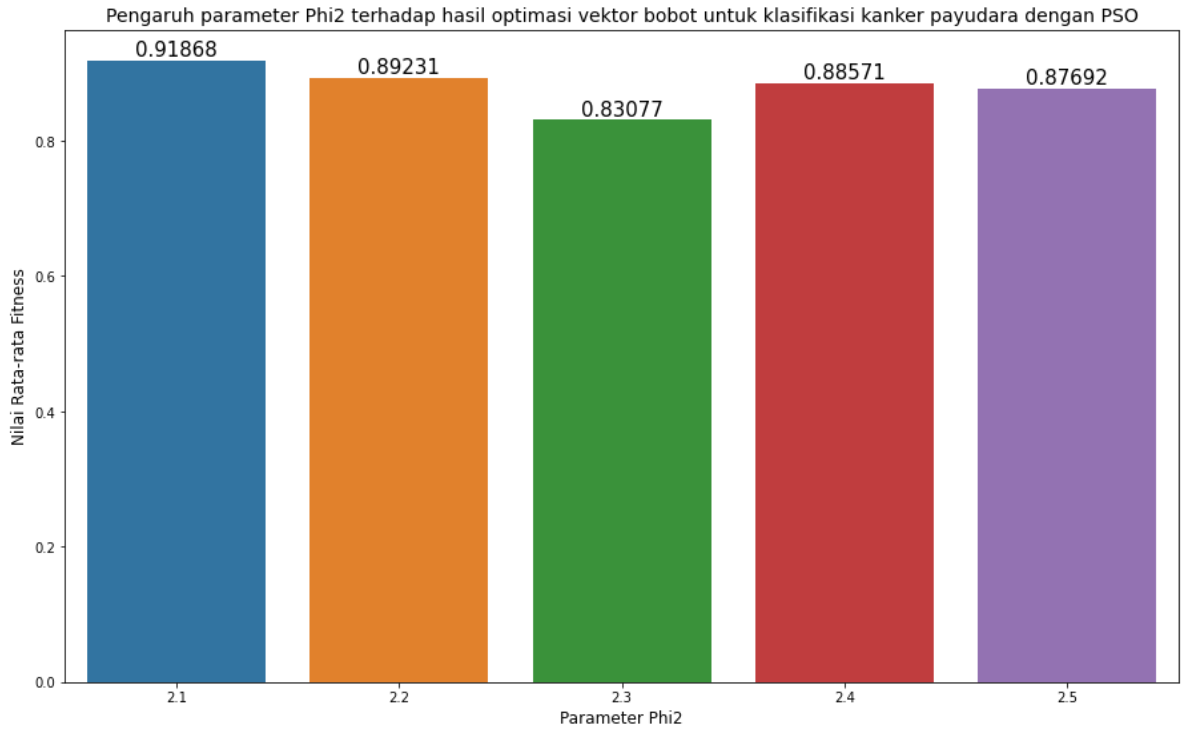
```
plt.figure(figsize=(15,9))
df_phi1 = pso_tuning_parameters_results[
    (pso_tuning_parameters_results['phi2'] == optimal_phi2) &
    (pso_tuning_parameters_results['inertia'] == optimal_inertia) &
    (pso_tuning_parameters_results['n_particles'] == optimal_n_particles)
][['phi1', 'mean_accuracy']]
list_phi1 = df_phi1['phi1']
mean_accuracy_phi1 = df_phi1['mean_accuracy']
plt.plot(df_phi1.set_index('phi1'), 'g-x')
plt.xticks(ticks=list_phi1)
plt.ylabel('Nilai Rata-rata Fitness', size=12)
plt.xlabel('Parameter Phi1', size=12)
plt.title('Pengaruh parameter Phi1 terhadap hasil optimasi vektor bobot untuk klasifikasi k

#zip phi1 and mean accuracy for annotate graph
for phi1, mean_accuracy in zip(list_phi1, mean_accuracy_phi1):
    label = "{:.3f}".format(mean_accuracy)
    plt.annotate(
        label,
        (phi1,mean_accuracy),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )
plt.show()
```



6.2 Analisis Pengaruh Parameter Phi2

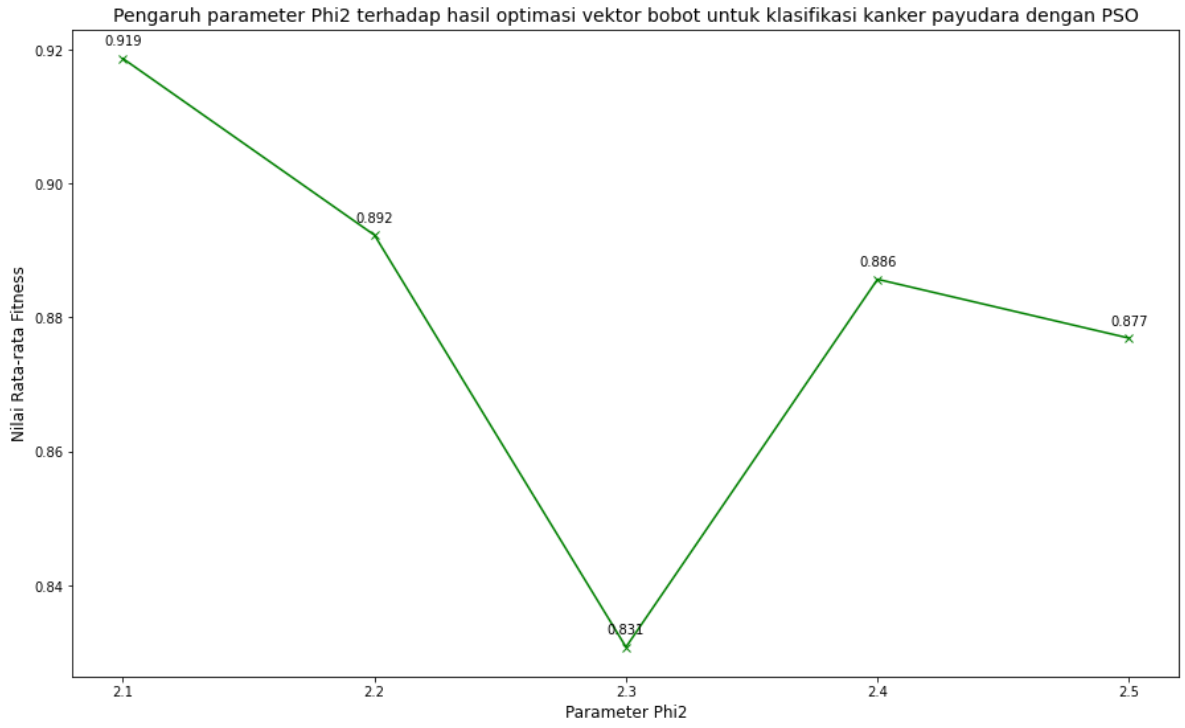
In [35]:

```
df_phi2 = pso_tuning_parameters_results[
    (pso_tuning_parameters_results['phi1'] == optimal_phi1) &
    (pso_tuning_parameters_results['inertia']==optimal_inertia)&
    (pso_tuning_parameters_results['n_particles'] == optimal_n_particles)
][['phi2','mean_accuracy']]

# Defining the plot size
plt.figure(figsize=(15, 9))
plots = sns.barplot(x="phi2", y="mean_accuracy", data=df_phi2)
plt.ylabel('Nilai Rata-rata Fitness', size=12)
plt.xlabel('Parameter Phi2', size=12)
plt.title('Pengaruh parameter Phi2 terhadap hasil optimasi vektor bobot untuk klasifikasi k
for bar in plots.patches:
    # Using Matplotlib's annotate function and
    # passing the coordinates where the annotation shall be done
    # x-coordinate: bar.get_x() + bar.get_width() / 2
    # y-coordinate: bar.get_height()
    # free space to be left to make graph pleasing: (0, 8)
    # ha and va stand for the horizontal and vertical alignment
    plots.annotate(format(bar.get_height(), '.5f'),
                   (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                   ha='center',
                   va='center',
                   size=15,
                   xytext=(0, 8),
                   textcoords='offset points')
plt.savefig('imageplot/phi2.jpg')
plt.show()
```



In [36]:

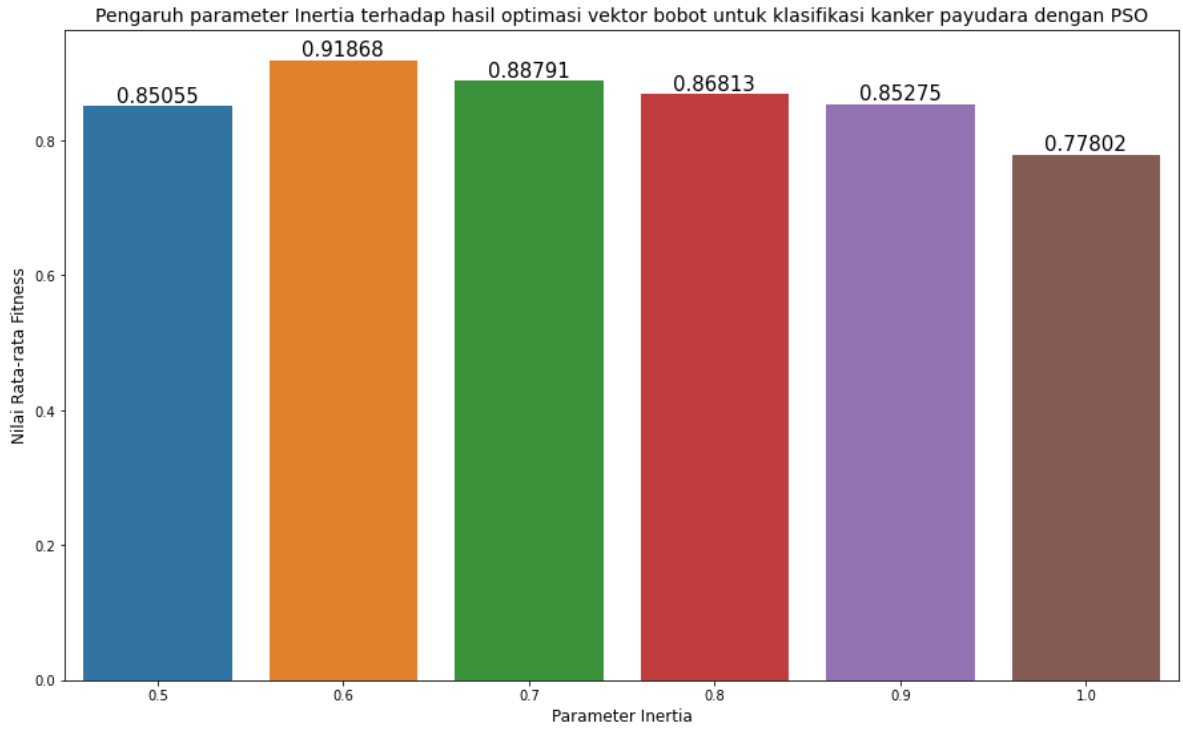
```
plt.figure(figsize=(15,9))

list_phi2 = df_phi2['phi2']
mean_accuracy_phi2 = df_phi2['mean_accuracy']
plt.plot(df_phi2.set_index('phi2'), 'g-x')
plt.xticks(ticks=list_phi2)
plt.ylabel('Nilai Rata-rata Fitness', size=12)
plt.xlabel('Parameter Phi2', size=12)
plt.title('Pengaruh parameter Phi2 terhadap hasil optimasi vektor bobot untuk klasifikasi k

#zip phi2 and mean accuracy for annotate graph
for phi2, mean_accuracy in zip(list_phi2, mean_accuracy_phi2):
    label = "{:.3f}".format(mean_accuracy)
    plt.annotate(
        label,
        (phi2,mean_accuracy),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )
plt.show()
```



6.3 Analisis Pengaruh Parameter Inertia

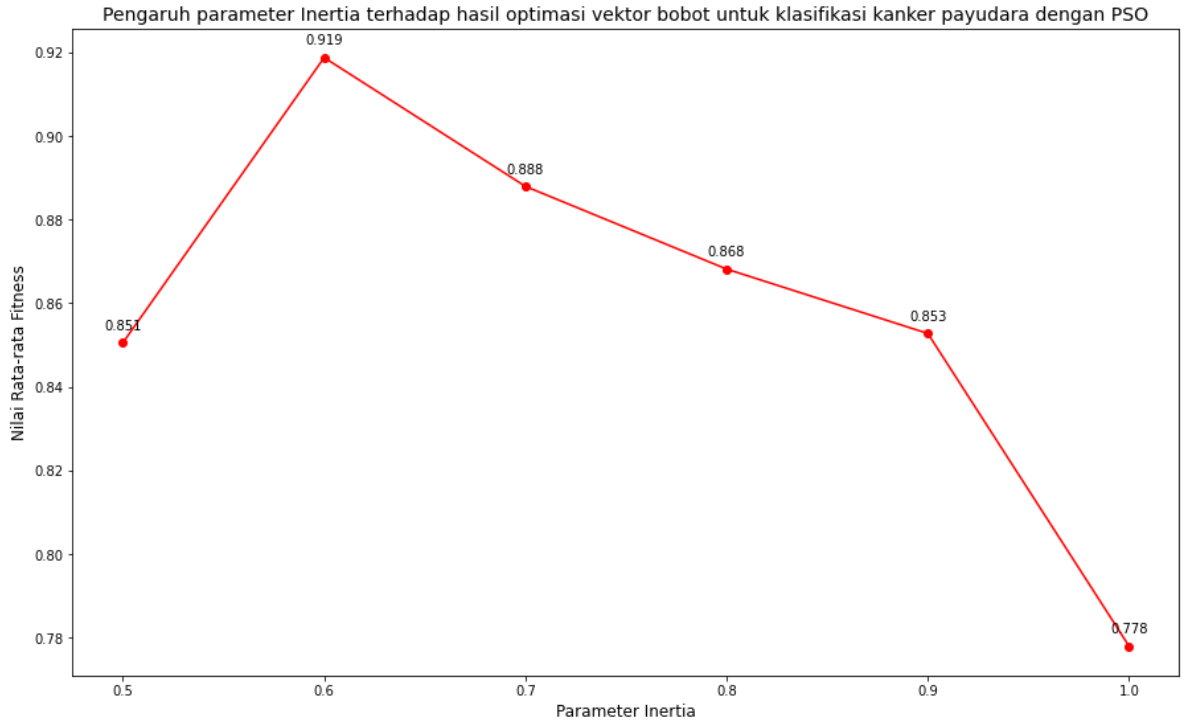
In [37]:

```
df_inertia = pso_tuning_parameters_results[
    (pso_tuning_parameters_results['phi1'] == optimal_phi1) &
    (pso_tuning_parameters_results['phi2'] == optimal_phi2)&
    (pso_tuning_parameters_results['n_particles']==optimal_n_particles)
][['inertia','mean_accuracy']]

# Defining the plot size
plt.figure(figsize=(15, 9))
plots = sns.barplot(x="inertia", y="mean_accuracy", data=df_inertia)
plt.ylabel('Nilai Rata-rata Fitness', size=12)
plt.xlabel('Parameter Inertia', size=12)
plt.title('Pengaruh parameter Inertia terhadap hasil optimasi vektor bobot untuk klasifikas
for bar in plots.patches:
    # Using Matplotlib's annotate function and
    # passing the coordinates where the annotation shall be done
    # x-coordinate: bar.get_x() + bar.get_width() / 2
    # y-coordinate: bar.get_height()
    # free space to be left to make graph pleasing: (0, 8)
    # ha and va stand for the horizontal and vertical alignment
    plots.annotate(format(bar.get_height(), '.5f'),
                   (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                   ha='center',
                   va='center',
                   size=15,
                   xytext=(0, 8),
                   textcoords='offset points')
plt.savefig('imageplot/inertia.jpg')
plt.show()
```



In [38]:

```
plt.figure(figsize=(15,9))
df_inertia = pso_tuning_parameters_results[
    (pso_tuning_parameters_results['phi1'] == optimal_phi1) &
    (pso_tuning_parameters_results['phi2'] == optimal_phi2)&
    (pso_tuning_parameters_results['n_particles']==optimal_n_particles)
][['inertia','mean_accuracy']]
list_inertia = df_inertia['inertia']
mean_accuracy_inertia = df_inertia['mean_accuracy']
plt.plot(df_inertia.set_index('inertia'), 'r-o')
plt.xticks(ticks=list_inertia)
plt.ylabel('Nilai Rata-rata Fitness', size=12)
plt.xlabel('Parameter Inertia', size=12)
plt.title('Pengaruh parameter Inertia terhadap hasil optimasi vektor bobot untuk klasifikas

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



In [ ]:

In [ ]:

```
```