```
pip install pyswarm
```

```
Requirement already satisfied: pyswarm in /usr/local/lib/python3.10/dist-packages (0.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from pyswarm) (1.25.2)
ERROR: Operation cancelled by user
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import lightgbm as lgb
from sklearn.metrics import accuracy_score
from pyswarm import pso
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

+ Code + Text

```
# Load the dataset
diabetes_data = pd.read_csv("/content/diabetes.csv")
```

```
# Split the dataset into features and target
X = diabetes_data.drop('Outcome', axis=1)
y = diabetes_data['Outcome']
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# SVM
svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_pred)
```

```
# Logistic Regression
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_pred)
```

```
# LightGBM
lgb_train = lgb.Dataset(X_train, y_train)
lgb_test = lgb.Dataset(X_test, y_test)
params = {
    'objective': 'binary',
    'metric': 'binary_error',
    'verbosity': -1
}
lgb_model = lgb.train(params, lgb_train, num_boost_round=100)
lgb_pred = lgb_model.predict(X_test)
lgb_pred_binary = [1 if x >= 0.5 else 0 for x in lgb_pred]
lgb_accuracy = accuracy_score(y_test, lgb_pred_binary)
```

```
import numpy as np

def objective_function(weights, X, y):
    weighted_sum = (weights[0] * np.array(svm_pred)) + (weights[1] * np.array(lr_pred)) + (weights[2] * np.array(lgb_pred_binary))
    weighted_sum_binary = [1 if x >= 0.5 else 0 for x in weighted_sum]
    return -accuracy_score(y, weighted_sum_binary)

lb = [0, 0, 0]
ub = [1, 1, 1]
weights, _ = pso(objective_function, lb, ub, args=(X_test, y_test))
```

```
Stopping search: maximum iterations reached --> 100
```

```
# Weighted ensemble prediction
weighted_sum = (weights[0] * np.array(svm_pred)) + (weights[1] * np.array(lr_pred)) + (weights[2] * np.array(lgb_pred_binary))
weighted_sum_binary = [1 if x >= 0.5 else 0 for x in weighted_sum]
ensemble_accuracy = accuracy_score(y_test, weighted_sum_binary)


print("SVM Accuracy:", svm_accuracy)
print("Logistic Regression Accuracy:", lr_accuracy)
print("LightGBM Accuracy:", lgb_accuracy)
print("Ensemble Accuracy (PSO-optimized):", ensemble_accuracy)
```

```
SVM Accuracy: 0.7337662337662337
Logistic Regression Accuracy: 0.7532467532467533
LightGBM Accuracy: 0.7077922077922078
Ensemble Accuracy (PSO-optimized): 0.7792207792207793
```

```
accuracies = {
    'SVM': svm_accuracy,
    'Logistic Regression': lr_accuracy,
    'LightGBM': lgb_accuracy,
    'Ensemble (PSO-optimized)': ensemble_accuracy
}
best_model = max(accuracies, key=accuracies.get)
print("\nBest Model:", best_model, "with Accuracy:", accuracies[best_model])
```

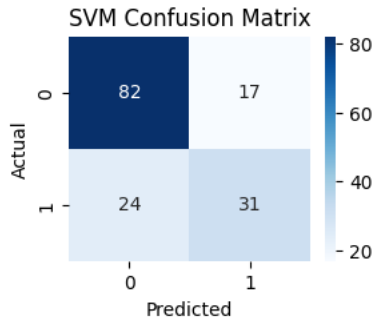
```
Best Model: Ensemble (PSO-optimized) with Accuracy: 0.7792207792207793
```

```
# Confusion matrices
plt.figure(figsize=(16, 12))
```

```
<Figure size 1600x1200 with 0 Axes>
<Figure size 1600x1200 with 0 Axes>
```

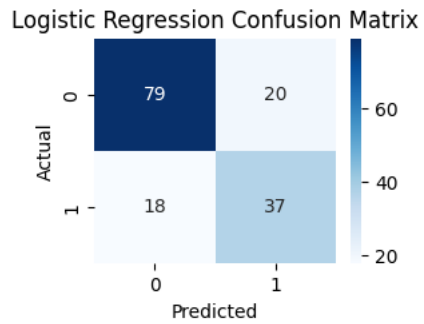
```
plt.subplot(2, 2, 1)
svm_cm = confusion_matrix(y_test, svm_pred)
sns.heatmap(svm_cm, annot=True, fmt="d", cmap="Blues")
plt.title("SVM Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

```
Text(50.722222222222214, 0.5, 'Actual')
```



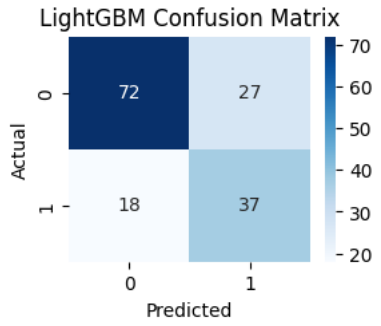

```
plt.subplot(2, 2, 2)
lr_cm = confusion_matrix(y_test, lr_pred)
sns.heatmap(lr_cm, annot=True, fmt="d", cmap="Blues")
plt.title("Logistic Regression Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

```
Text(321.26767676767673, 0.5, 'Actual')
```

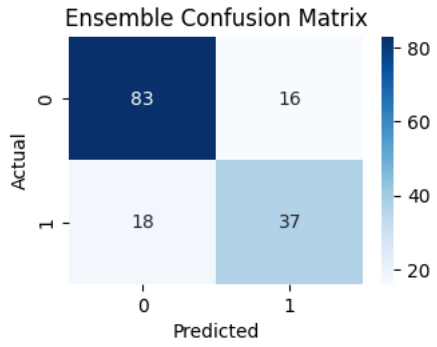


```
plt.subplot(2, 2, 3)
lgb_cm = confusion_matrix(y_test, lgb_pred_binary)
sns.heatmap(lgb_cm, annot=True, fmt="d", cmap="Blues")
plt.title("LightGBM Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

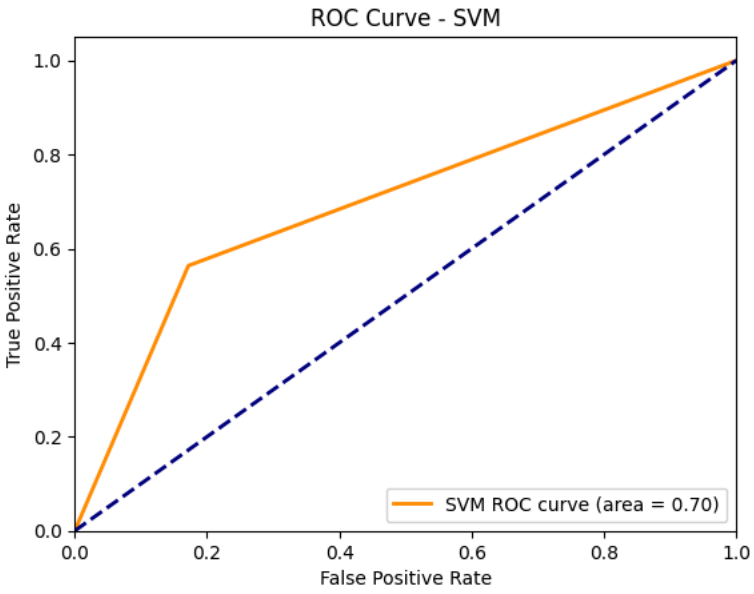
```
Text(50.722222222222214, 0.5, 'Actual')
```

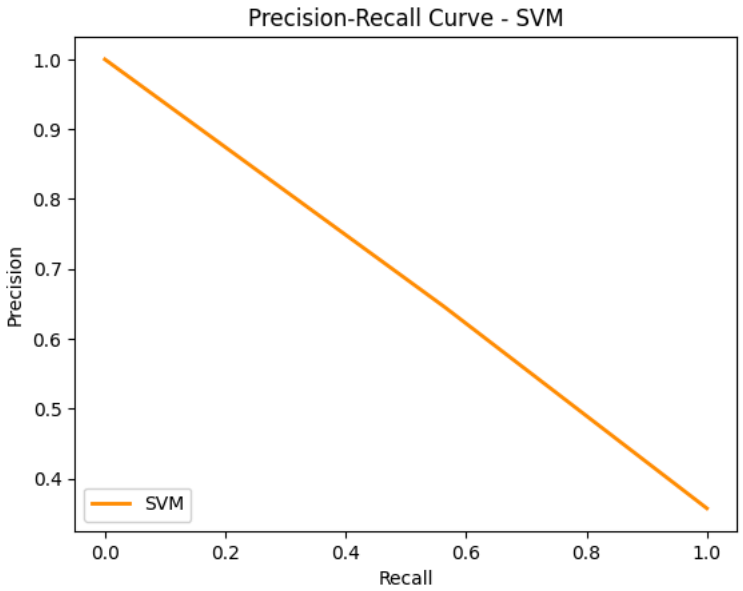



```
plt.subplot(2, 2, 4)
ensemble_cm = confusion_matrix(y_test, weighted_sum_binary)
sns.heatmap(ensemble_cm, annot=True, fmt="d", cmap="Blues")
plt.title("Ensemble Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")

plt.tight_layout()
plt.show()
```

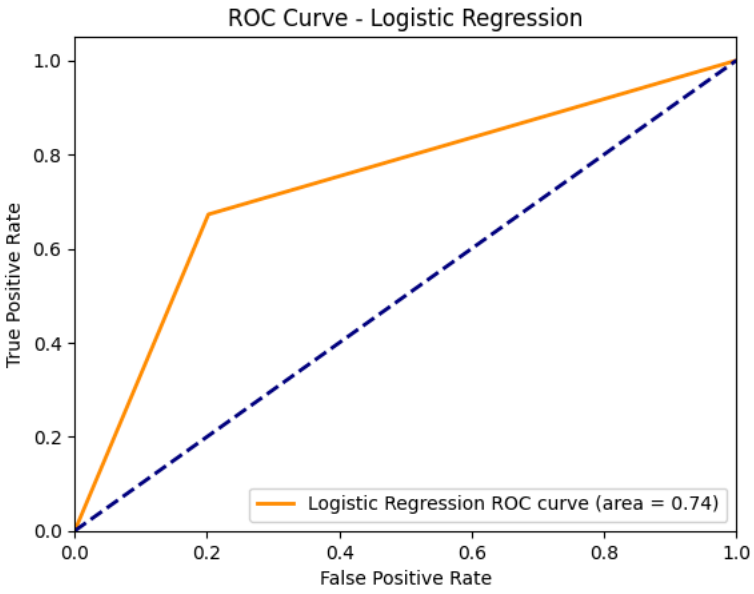


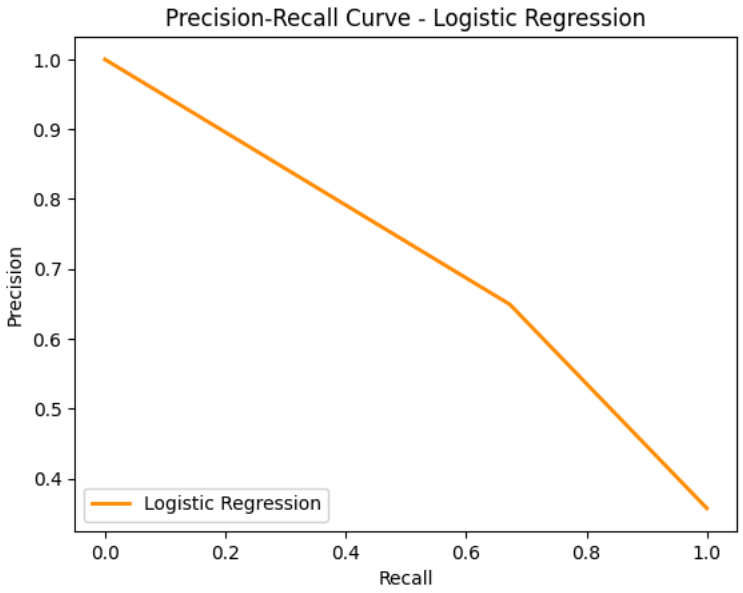

```
from sklearn.metrics import roc_curve, precision_recall_curve, auc

# ROC Curve for SVM
svm_fpr, svm_tpr, _ = roc_curve(y_test, svm_pred)
svm_auc = auc(svm_fpr, svm_tpr)
plt.figure()
plt.plot(svm_fpr, svm_tpr, color='darkorange', lw=2, label='SVM ROC curve (area = %0.2f)' % svm_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - SVM')
plt.legend(loc="lower right")
plt.show()

# Precision-Recall Curve for SVM
svm_precision, svm_recall, _ = precision_recall_curve(y_test, svm_pred)
plt.figure()
plt.plot(svm_recall, svm_precision, color='darkorange', lw=2, label='SVM')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve - SVM')
plt.legend(loc="lower left")
plt.show()
```



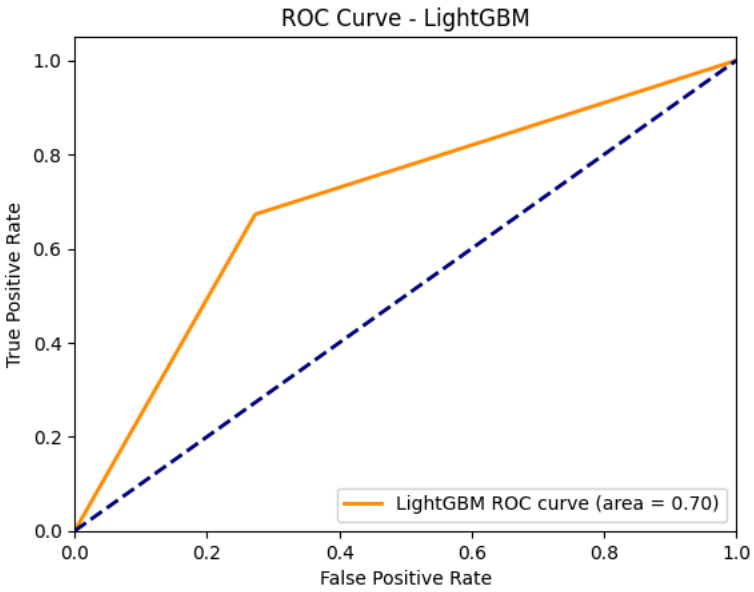


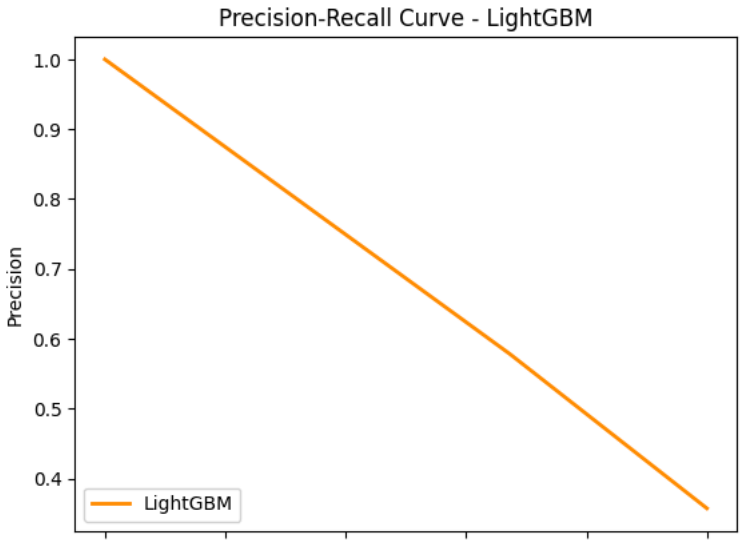

```
# ROC Curve for Logistic Regression
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_pred)
lr_auc = auc(lr_fpr, lr_tpr)
plt.figure()
plt.plot(lr_fpr, lr_tpr, color='darkorange', lw=2, label='Logistic Regression ROC curve (area = %0.2f)' % lr_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend(loc="lower right")
plt.show()

# Precision-Recall Curve for Logistic Regression
lr_precision, lr_recall, _ = precision_recall_curve(y_test, lr_pred)
plt.figure()
plt.plot(lr_recall, lr_precision, color='darkorange', lw=2, label='Logistic Regression')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve - Logistic Regression')
plt.legend(loc="lower left")
plt.show()
```






```
# ROC Curve for LightGBM
lgb_fpr, lgb_tpr, _ = roc_curve(y_test, lgb_pred_binary)
lgb_auc = auc(lgb_fpr, lgb_tpr)
plt.figure()
plt.plot(lgb_fpr, lgb_tpr, color='darkorange', lw=2, label='LightGBM ROC curve (area = %0.2f)' % lgb_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - LightGBM')
plt.legend(loc="lower right")
plt.show()

# Precision-Recall Curve for LightGBM
lgb_precision, lgb_recall, _ = precision_recall_curve(y_test, lgb_pred_binary)
plt.figure()
plt.plot(lgb_recall, lgb_precision, color='darkorange', lw=2, label='LightGBM')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve - LightGBM')
plt.legend(loc="lower left")
plt.show()
```


ROC Curve - LightGBM
True Positive Rate
False Positive Rate
LightGBM ROC curve (area = 0.70)
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall Curve - LightGBM
Precision
LightGBM
0.4
0.5
0.6
0.7
0.8
0.9
1.0