In [3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler , LabelEncoder
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

In [5]:
```
#load dataset
iphone=pd.read_csv(r"C:\Users\mamun\OneDrive\iphone_purchase_records.csv")
iphone.head()
```

Out[5]:

	Gender	Age	Salary	Purchase Iphone
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0

In [6]:
```
#encode the 'Gender' column
label = LabelEncoder()
iphone['Gender']=label.fit_transform(iphone['Gender'])
iphone.head()
```

Out[6]:

	Gender	Age	Salary	Purchase Iphone
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0

In [7]:
```
#separate features and target variable
X = iphone[['Gender','Age','Salary']]
y = iphone['Purchase Iphone']
```

In [8]:
```
#splitting data into tarining and testing set
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state
```

In [9]:
```
#feature scaling
scaler=StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [10]:
```
#initialize model
models = {
    "SVM": SVC(kernel='linear',random_state=0),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "Logistic Regression": LogisticRegression(random_state=0)
}
```

In [12]:
```
#train, predict and evaluate each model
for name,model in models.items():
    print(f"\nModel:{name}")

    #train the model
    model.fit(X_train,y_train)

    #make predictions
    y_pred=model.predict(X_test)

    #calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy:{accuracy * 100:.2f}%")

    #confusion matrix and classification report
    cm= confusion_matrix(y_test,y_pred)
    report = classification_report(y_test,y_pred)
    print("confusion_matrix:\n",cm)
    print("classification_report:\n",report)

    #print correct and incorrect predictions
    correct_predictions = [(i,pred) for i,(pred,true) in enumerate(zip(y_pred,
    incorrect_predictions = [(i,pred) for i,(pred,true) in enumerate(zip(y_pre

    print("Correct predictions:",correct_predictions)
    print("\nIncorrect predictions:",incorrect_predictions)
```

```
Model:SVM
Accuracy:89.00%
confusion_matrix:
 [[66  2]
 [ 9 23]]
classification_report:
               precision    recall  f1-score   support

           0       0.88      0.97      0.92        68
           1       0.92      0.72      0.81        32

    accuracy                           0.89       100
   macro avg       0.90      0.84      0.87       100
weighted avg       0.89      0.89      0.89       100

Correct predictions: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0),
(7, 1), (8, 0), (9, 0), (10, 0), (11, 0), (12, 0), (13, 0), (14, 0), (15, 0),
(16, 0), (17, 0), (18, 1), (19, 0), (20, 0), (21, 1), (22, 0), (23, 1), (24,
0), (25, 1), (26, 0), (27, 0), (28, 0), (29, 0), (30, 0), (32, 1), (33, 0),
(34, 0), (35, 0), (36, 0), (37, 0), (38, 0), (40, 0), (41, 0), (42, 0), (43,
0), (44, 1), (45, 0), (46, 0), (47, 1), (48, 0), (49, 1), (50, 1), (51, 0),
(52, 0), (53, 0), (54, 1), (56, 0), (57, 0), (59, 0), (60, 0), (61, 1), (62,
0), (64, 0), (65, 1), (66, 0), (67, 0), (68, 0), (69, 0), (70, 1), (71, 0),
(72, 0), (74, 0), (75, 0), (77, 0), (78, 1), (79, 1), (80, 1), (82, 0), (83,
0), (84, 1), (85, 1), (86, 0), (87, 1), (89, 0), (90, 0), (91, 1), (92, 0),
(93, 0), (94, 0), (96, 0), (98, 1), (99, 1)]

Incorrect predictions: [(31, 0), (39, 0), (55, 0), (58, 0), (63, 0), (73, 0),
(76, 1), (81, 1), (88, 0), (95, 0), (97, 0)]

Model:KNN
Accuracy:93.00%
confusion_matrix:
 [[64  4]
 [ 3 29]]
classification_report:
               precision    recall  f1-score   support

           0       0.96      0.94      0.95        68
           1       0.88      0.91      0.89        32

    accuracy                           0.93       100
   macro avg       0.92      0.92      0.92       100
weighted avg       0.93      0.93      0.93       100

Correct predictions: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0),
(7, 1), (8, 0), (10, 0), (11, 0), (12, 0), (13, 0), (14, 0), (16, 0), (17,
0), (18, 1), (19, 0), (20, 0), (21, 1), (22, 0), (23, 1), (24, 0), (25, 1),
(26, 0), (27, 0), (28, 0), (29, 0), (30, 0), (32, 1), (33, 0), (34, 0), (35,
0), (36, 0), (37, 0), (38, 0), (39, 1), (40, 0), (41, 0), (42, 0), (43, 0),
(44, 1), (45, 0), (46, 0), (47, 1), (48, 0), (49, 1), (50, 1), (51, 0), (52,
0), (54, 1), (55, 1), (56, 0), (57, 0), (58, 1), (59, 0), (60, 0), (61, 1),
(62, 0), (63, 1), (64, 0), (65, 1), (66, 0), (67, 0), (68, 0), (69, 0), (70,
1), (71, 0), (72, 0), (73, 1), (74, 0), (75, 0), (76, 0), (77, 0), (78, 1),
(79, 1), (80, 1), (82, 0), (83, 0), (84, 1), (86, 0), (87, 1), (88, 1), (89,
0), (90, 0), (91, 1), (92, 0), (93, 0), (94, 0), (96, 0), (97, 1), (98, 1),
```

```
(99, 1)]

Incorrect predictions: [(9, 1), (15, 1), (31, 0), (53, 1), (81, 1), (85, 0),
(95, 0)]

Model:Logistic Regression
Accuracy:90.00%
confusion_matrix:
 [[65  3]
 [ 7 25]]
classification_report:
               precision    recall  f1-score   support

           0       0.90      0.96      0.93        68
           1       0.89      0.78      0.83        32

    accuracy                           0.90       100
   macro avg       0.90      0.87      0.88       100
weighted avg       0.90      0.90      0.90       100

Correct predictions: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0),
(7, 1), (8, 0), (10, 0), (11, 0), (12, 0), (13, 0), (14, 0), (15, 0), (16,
0), (17, 0), (18, 1), (19, 0), (20, 0), (21, 1), (22, 0), (23, 1), (24, 0),
(25, 1), (26, 0), (27, 0), (28, 0), (29, 0), (30, 0), (32, 1), (33, 0), (34,
0), (35, 0), (36, 0), (37, 0), (38, 0), (39, 1), (40, 0), (41, 0), (42, 0),
(43, 0), (44, 1), (45, 0), (46, 0), (47, 1), (48, 0), (49, 1), (50, 1), (51,
0), (52, 0), (53, 0), (54, 1), (56, 0), (57, 0), (58, 1), (59, 0), (60, 0),
(61, 1), (62, 0), (64, 0), (65, 1), (66, 0), (67, 0), (68, 0), (69, 0), (70,
1), (71, 0), (72, 0), (74, 0), (75, 0), (77, 0), (78, 1), (79, 1), (80, 1),
(82, 0), (83, 0), (84, 1), (85, 1), (86, 0), (87, 1), (89, 0), (90, 0), (91,
1), (92, 0), (93, 0), (94, 0), (96, 0), (98, 1), (99, 1)]

Incorrect predictions: [(9, 1), (31, 0), (55, 0), (63, 0), (73, 0), (76, 1),
(81, 1), (88, 0), (95, 0), (97, 0)]
```

In [ ]: