In [6]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc


from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier


import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
df = pd.read_csv("C:/Users/HP/Desktop/spotify_churn_dataset.csv")
```

In [8]:
```
df.head()
```

Out[8]:

	user_id	gender	age	country	subscription_type	listening_time	songs_played_per_day
0	1	Female	54	CA	Free	26	23
1	2	Other	33	DE	Family	141	62
2	3	Male	38	AU	Premium	199	38
3	4	Female	22	CA	Student	36	2
4	5	Other	29	US	Family	250	57

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8000 entries, 0 to 7999
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   user_id                8000 non-null   int64  
 1   gender                 8000 non-null   object 
 2   age                    8000 non-null   int64  
 3   country                8000 non-null   object 
 4   subscription_type      8000 non-null   object 
 5   listening_time         8000 non-null   int64  
 6   songs_played_per_day   8000 non-null   int64  
 7   skip_rate              8000 non-null   float64
 8   device_type            8000 non-null   object 
 9   ads_listened_per_week  8000 non-null   int64  
 10  offline_listening      8000 non-null   int64  
 11  is_churned             8000 non-null   int64  
dtypes: float64(1), int64(7), object(4)
memory usage: 750.1+ KB
```

In [10]:
```
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	r
user_id	8000.0	4000.500000	2309.545410	1.0	2000.75	4000.5	6000.25	80
age	8000.0	37.662125	12.740359	16.0	26.00	38.0	49.00	
listening_time	8000.0	154.068250	84.015596	10.0	81.00	154.0	227.00	2
songs_played_per_day	8000.0	50.127250	28.449762	1.0	25.00	50.0	75.00	
skip_rate	8000.0	0.300127	0.173594	0.0	0.15	0.3	0.45	
ads_listened_per_week	8000.0	6.943875	13.617953	0.0	0.00	0.0	5.00	
offline_listening	8000.0	0.747750	0.434331	0.0	0.00	1.0	1.00	
is_churned	8000.0	0.258875	0.438044	0.0	0.00	0.0	1.00	

In [11]:
```
df.shape
```

Out[11]:
```
(8000, 12)
```

In [12]:
```
target = "is_churned"
```

In [13]:
```
print(df.columns.tolist())
```

```
['user_id', 'gender', 'age', 'country', 'subscription_type', 'listening_time', 'song
s_played_per_day', 'skip_rate', 'device_type', 'ads_listened_per_week', 'offline_lis
tening', 'is_churned']
```

In [14]:
```
le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = le.fit_transform(df[col])
```

In [15]:
```
X = df.drop('is_churned', axis=1)  # Replace with your target column name if differ
y = df['is_churned']
```

In [16]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran
```

In [18]:
```
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(eval_metric='logloss', use_label_encoder=False, random
    "Naive Bayes": GaussianNB(),
    "SVM": SVC(kernel='rbf', probability=True, random_state=42),
    "KNN": KNeighborsClassifier()
}
```

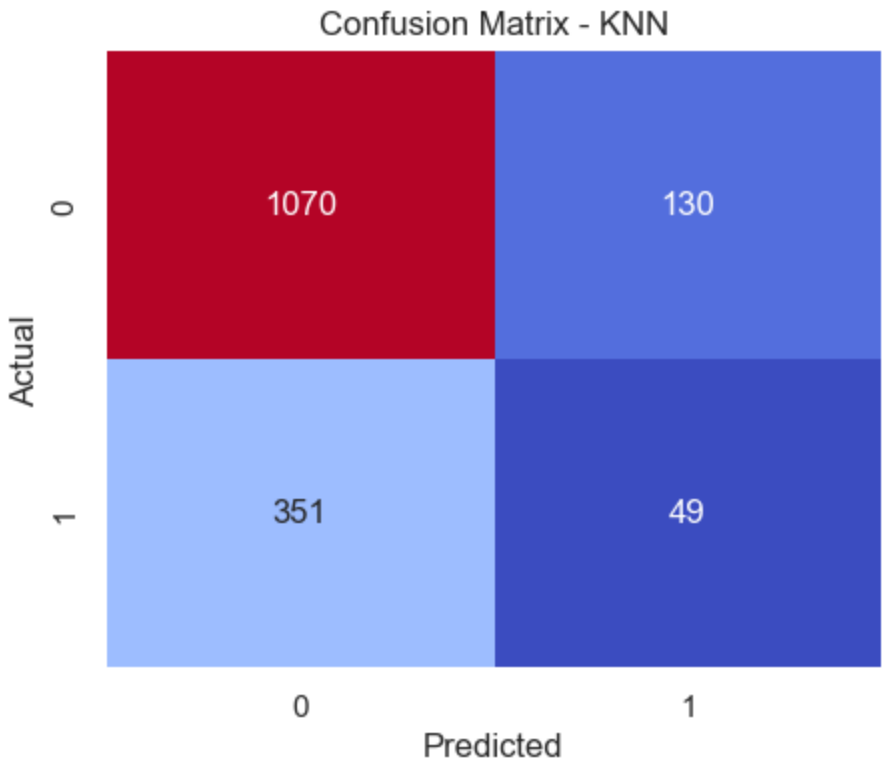
In [19]:
```
results = {}
auc_results = {}
conf_matrices = {}
roc_data = {}
```

In [20]:
```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]  # For ROC & AUC
```

In [21]:
```
results[name] = accuracy_score(y_test, y_pred)
conf_matrices[name] = confusion_matrix(y_test, y_pred)
```

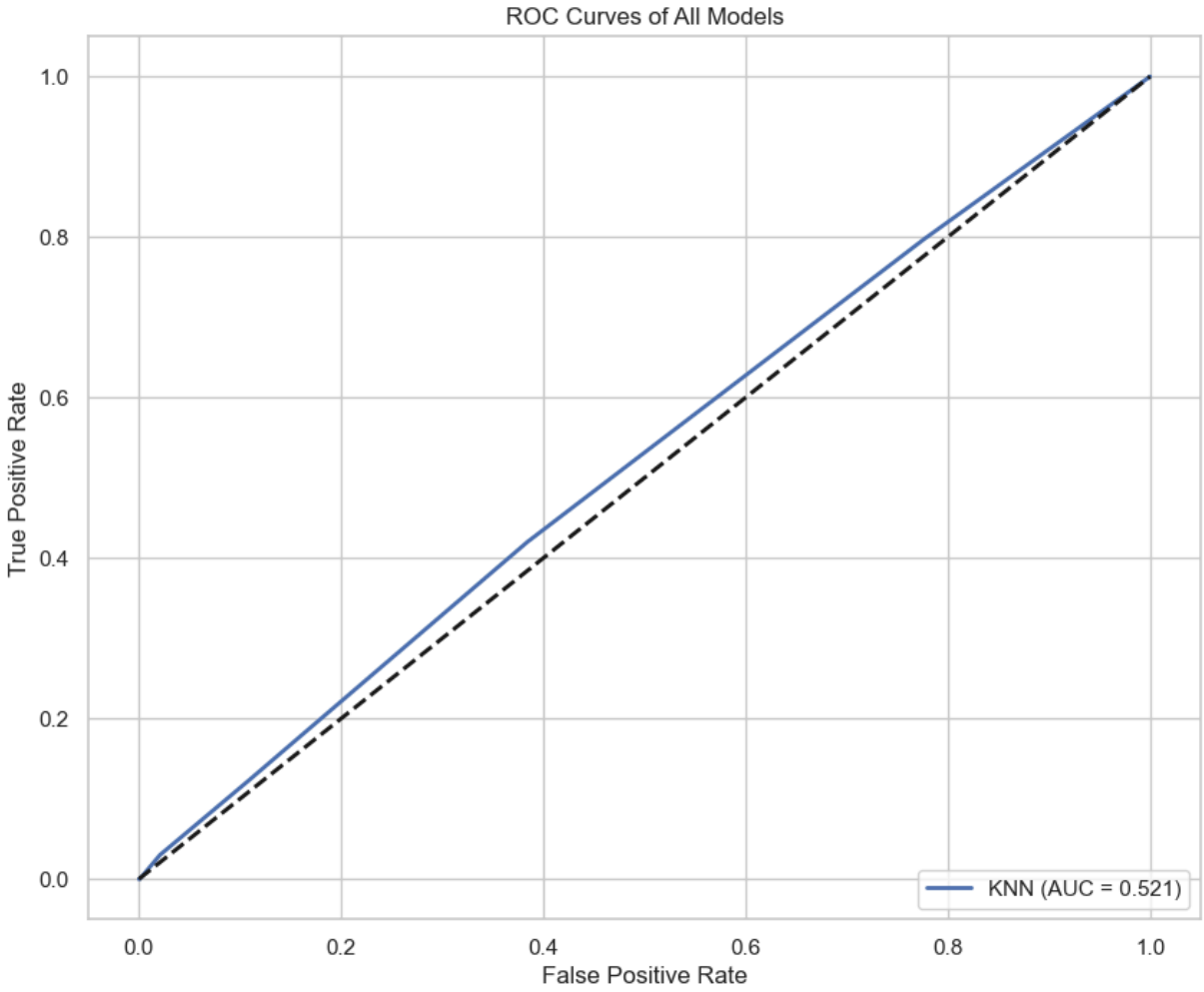
In [22]:
```
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
roc_data[name] = (fpr, tpr, roc_auc)
auc_results[name] = roc_auc
```

In [25]:
```
for name, cm in conf_matrices.items():
    plt.figure(figsize=(5,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='coolwarm', cbar=False)
    plt.title(f'Confusion Matrix - {name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```



In [26]:
```
plt.figure(figsize=(10,8))
for name, (fpr, tpr, roc_auc) in roc_data.items():
    plt.plot(fpr, tpr, lw=2, label=f'{name} (AUC = {roc_auc:.3f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.title('ROC Curves of All Models')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```


ROC Curves of All Models
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
KNN (AUC = 0.521)

In [ ]: