In [100]:

```
import pandas as pd
import numpy as np
```

In [101]:

```
df_mail_rishab=pd.read_csv('C:\\Users\\Rishab\\Downloads\\Rishab_ML.csv')
```

In [102]:

```
df_mail_sushant=pd.read_csv('C:\\Users\\Rishab\\Downloads\\Sushant_ML.csv')
```

In [103]:

```
df_mail_shruti=pd.read_csv('C:\\Users\\Rishab\\Downloads\\Shruti_ML.csv')
```

In [104]:

```
df_mail_rishab['Label'].unique()   # 5 labels
```

Out[104]:

```
array(['Updates', 'Social', 'Promotion', 'Forum', 'Spam'], dtype=objec
t)
```

In [105]:

```
df_mail_sushant['Label'].unique()
```

Out[105]:

```
array(['Update', 'Social', 'Promotion', 'Forum', 'Span'], dtype=objec
t)
```

In [106]:

```
df_mail_shruti['Label'].unique()
```

Out[106]:

```
array(['Span', 'Forum', 'Update', 'Social', 'Promotion'], dtype=objec
t)
```

In [107]:

```
df_mail_rishab.head()   # Dataset has been scraped, the subject , body of email and
```

Out[107]:

| | Subject | Body | Label |
|---|---|---|---|
| 0 | Security alert | \t\n\t <https://www.gstatic.com/images/brandin... | Updates |
| 1 | Trade SOL and Share $150,000 in SOL! | <https://www.binance.com/bapi/composite/v1/pu... | Updates |
| 2 | Your sign-in has changed | \t\n\t <https://www.gstatic.com/images/brandin... | Updates |
| 3 | Make Your Stablecoins More Stable! | <https://public.bnbstatic.com/image/ufo/20211... | Updates |
| 4 | The 2% Theory community is waiting for you! | Hello,\n\nOne of the best parts of 2% Theory i... | Updates |

In [108]:

```
df_mail_sushant.head()
```

Out[108]:

| | Subject | Body | Label |
|---|---|---|---|
| 0 | Why are BITS Hyderabad students protesting aga... | I don’t think any more explanation is needed a... | Update |
| 1 | Review your latest connections | See Atreya's and other people's connections, e... | Update |
| 2 | Grab your rewards on Cloud DevJam | \t\n <https://www.techgig.com/files/contest_up... | Update |
| 3 | Data Scientist Interview Process at Microsoft ... | <https://medium.com/_/stat?event=email.opened... | Update |
| 4 | [GitHub Education Community] Summary | A brief summary since your last visit on Novem... | Update |

In [109]:

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()      #Labels have been encoded
df_mail_rishab['Label'] = label_encoder.fit_transform(df_mail_rishab['Label'])
df_mail_sushant['Label'] = label_encoder.fit_transform(df_mail_sushant['Label'])
```

In [110]:

```
df_mail_shruti['Label'] = label_encoder.fit_transform(df_mail_shruti['Label'])
```

In [111]:

```
df_mail_rishab['Label'].unique()
```

Out[111]:

```
array([4, 2, 1, 0, 3])
```

In [112]:

```
df_mail_sushant['Label'].unique()
```

Out[112]:

```
array([4, 2, 1, 0, 3])
```

In [113]:

```
df_mail_sushant['Label'].unique()
```

Out[113]:

```
array([4, 2, 1, 0, 3])
```

In [114]:

```
var_rishab=pd.DataFrame(df_mail_rishab['Subject'] + ' ' +df_mail_rishab['Body'])
var_rishab
```

Out[114]:

| | 0 |
|---|---|
| 0 | Security alert \t\n\t <https://www.gstatic.com... |
| 1 | Trade SOL and Share $150,000 in SOL! <https:/... |
| 2 | Your sign-in has changed \t\n\t <https://www.g... |
| 3 | Make Your Stablecoins More Stable! <https://p... |
| 4 | The 2% Theory community is waiting for you! He... |
| ... | ... |
| 1391 | Confirmation,Receipt ! <https://d15k2d11r6t6rl... |
| 1392 | BIDEN: IRAS, 401(K)S ARE SAFE! AMERICANS WORRY... |
| 1393 | 🎉Celebrations Begin At Home 🏠 Make Your Happy ... |
| 1394 | Do you feel like fucking a sexy gal? Last seen... |
| 1395 | rishabkabdi99@gmail.com, Today's Offer: Flat 6... |

1396 rows × 1 columns

In [116]:

```
var_sushant=pd.DataFrame(df_mail_sushant['Subject'] + ' ' +df_mail_sushant['Body'])
```

In [117]:

```
var_shruti=pd.DataFrame(df_mail_shruti['Subject'] + ' ' +df_mail_shruti['Body'])
```

In [118]:

```
Y_rishab=pd.DataFrame(df_mail_rishab['Label'])
Y_rishab.describe()
```

Out[118]:

| | Label |
|---|---|
| **count** | 1396.000000 |
| **mean** | 2.188395 |
| **std** | 1.168724 |
| **min** | 0.000000 |
| **25%** | 1.000000 |
| **50%** | 2.000000 |
| **75%** | 3.000000 |
| **max** | 4.000000 |

In [119]:

```
Y_sushant=pd.DataFrame(df_mail_sushant['Label'])
Y_sushant.describe()
```

Out[119]:

| | Label |
|---|---|
| **count** | 1356.000000 |
| **mean** | 2.163717 |
| **std** | 1.394151 |
| **min** | 0.000000 |
| **25%** | 1.000000 |
| **50%** | 2.000000 |
| **75%** | 3.000000 |
| **max** | 4.000000 |

In [120]:

```
Y_shruti=pd.DataFrame(df_mail_shruti['Label'])
Y_shruti.describe()
```

Out[120]:

| | Label |
|---|---|
| **count** | 1365.000000 |
| **mean** | 2.127473 |
| **std** | 1.413648 |
| **min** | 0.000000 |
| **25%** | 1.000000 |
| **50%** | 2.000000 |
| **75%** | 3.000000 |
| **max** | 4.000000 |

In [121]:

```
X_rishab=var_rishab
```

In [122]:

```
X_sushant=var_sushant
```

In [123]:

```
X_shruti=var_shruti
```

In [124]:

```
X_rishab = X_rishab[0]
Y_rishab = Y_rishab.Label
```

In [125]:

```
X_sushant = X_sushant[0]
Y_sushant = Y_sushant.Label
```

In [126]:

```
X_shruti = X_shruti[0]
Y_shruti = Y_shruti.Label
```

In [127]:

```
from sklearn.model_selection import train_test_split
X_train_rishab, X_test_rishab, y_train_rishab, y_test_rishab = train_test_split(X_ri
X_train_sushant, X_test_sushant, y_train_sushant, y_test_sushant = train_test_split(
```

In [128]:

```
X_train_shruti, X_test_shruti, y_train_shruti, y_test_shruti = train_test_split(X_sh
```

In [129]:

```
X_train_rishab
```

Out[129]:

```
60      Alert: JoSAA, Seat Allocation (Round 3) \t\n <...
817     Weekly newsletter of Stores Domino - Issue #14...
589     Rishab Kabdi, you're getting noticed See who's...
104     Notification from Axis Bank Dear Customer, We ...
1138    I use dirty talk to get you what you want <htt...
                            ...
1147    rishabkabdi99, Play Now With 300% Welcome Bonu...
1344    Get your FREE LeafFilter Estimate now and save...
527     HCL Technologies is looking for: Data Scientis...
1149    confirmation <https://d15k2d11r6t6rl.cloudfron...
1289    rishabkabdi99@gmail.com, Book free* railway ti...
Name: 0, Length: 1116, dtype: object
```

In [130]:

```
X_train_rishab.head()
y_train_rishab.head()
X_train_rishab.describe()
```

Out[130]:

```
count                                                1116
unique                                               1116
top     🍀 🤠    Don't miss out on the Biggest Plant Di...
freq                                                    1
Name: 0, dtype: object
```

In [131]:

```
import re
def clean_text(text):
    text = text.lower()
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r"\'scuse", " excuse ", text)
    text = re.sub('\W', ' ', text)
    text = re.sub('\s+', ' ', text)
    text = re.sub(r'http\S+', '', text)

    text = text.strip('\t')
    text = text.strip('\n')
    text = text.strip('\\')

    text = text.strip(' ')


    return text
```

In [132]:

```
X_train_rishab=pd.DataFrame(X_train_rishab)
X_test_rishab=pd.DataFrame(X_test_rishab)
```

In [133]:

```
X_train_sushant=pd.DataFrame(X_train_sushant)
X_test_sushant=pd.DataFrame(X_test_sushant)
```

In [134]:

```
X_train_shruti=pd.DataFrame(X_train_shruti)
X_test_shruti=pd.DataFrame(X_test_shruti)
```

In [135]:

```
X_train_rishab[0] = X_train_rishab[0].map(lambda com : clean_text(com))    #Data cle
X_test_rishab[0] = X_test_rishab[0].map(lambda com : clean_text(com))
```

In [136]:

```
X_train_sushant[0] = X_train_sushant[0].map(lambda com : clean_text(com))    #Data c
X_test_sushant[0] = X_test_sushant[0].map(lambda com : clean_text(com))
```

In [137]:

```
X_train_shruti[0] = X_train_shruti[0].map(lambda com : clean_text(com))    #Data cle
X_test_shruti[0] = X_test_shruti[0].map(lambda com : clean_text(com))
```

In [138]:

```
X_train_rishab["Body"] = X_train_rishab[0]
X_train_sushant["Body"] = X_train_sushant[0]
X_train_rishab = X_train_rishab.drop(labels= 0, axis=1)

X_train_rishab
```

Out[138]:

| | Body |
|---|---|
| **60** | alert josaa seat allocation round 3 www shiks... |
| **817** | weekly newsletter of stores domino issue 14 de... |
| **589** | rishab kabdi you are getting noticed see who l... |
| **104** | notification from axis bank dear customer we w... |
| **1138** | i use dirty talk to get you what you want http... |
| **...** | ... |
| **1147** | rishabkabdi99 play now with 300 welcome bonus ... |
| **1344** | get your free leaffilter estimate now and save... |
| **527** | hcl technologies is looking for data scientist... |
| **1149** | confirmation d15k2d11r6t6rl cloudfront net pu... |
| **1289** | rishabkabdi99 gmail com book free railway tick... |

1116 rows × 1 columns

In [139]:

```
X_train_sushant = X_train_sushant.drop(labels= 0, axis=1)
```

In [140]:

```
X_train_shruti["Body"] = X_train_shruti[0]
X_train_shruti = X_train_shruti.drop(labels= 0, axis=1)
```

In [141]:

```
X_test_rishab["Body"] = X_test_rishab[0]
X_test_sushant["Body"] = X_test_sushant[0]
X_test_rishab = X_test_rishab.drop(labels= 0, axis=1)
X_test_sushant = X_test_sushant.drop(labels= 0, axis=1)
```

In [142]:

```
X_test_shruti["Body"] = X_test_shruti[0]
X_test_shruti = X_test_shruti.drop(labels= 0, axis=1)
```

In [143]:

```
y_train_sushant.value_counts()
```

Out[143]:

```
4    258
2    242
3    202
1    197
0    185
Name: Label, dtype: int64
```

In [144]:

```
type(X_train_rishab)
```

Out[144]:

```
pandas.core.frame.DataFrame
```

In [145]:

```
y_test_rishab.head()
```

Out[145]:

```
7       4
1048    0
326     2
564     2
689     2
Name: Label, dtype: int32
```

In [146]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
cv=TfidfVectorizer(ngram_range=(1,4))       # 1 gram, 2 gram, 3 gram and 4 gram used
cv.fit(X_train_rishab['Body'])
cv.fit(X_train_sushant['Body'])
```

Out[146]:

```
TfidfVectorizer(ngram_range=(1, 4))
```

In [147]:

```
cv.fit(X_train_shruti['Body'])
```

Out[147]:

```
TfidfVectorizer(ngram_range=(1, 4))
```

In [148]:

```
dtv_rishab = cv.transform(X_train_rishab['Body'])
dtv_sushant = cv.transform(X_train_sushant['Body'])
type(dtv_rishab)
```

Out[148]:

```
scipy.sparse.csr.csr_matrix
```

In [149]:

```
dtv_shruti = cv.transform(X_train_shruti['Body'])
```

In [150]:

```
dtv_rishab = dtv_rishab.toarray()
dtv_sushant = dtv_sushant.toarray()
```

In [151]:

```
dtv_shruti = dtv_shruti.toarray()
```

In [152]:

```
print(f"Rishab_Training Data Shape: {X_train_rishab['Body'].shape}\nRishab_Test Data
print(f"Sushant_Training Data Shape: {X_train_sushant['Body'].shape}\nSushant_Test D
print(f"Shruti_Training Data Shape: {X_train_shruti['Body'].shape}\nShruti_Test Data
```

```
Rishab_Training Data Shape: (1116,)
Rishab_Test Data Shape: (280,)
Sushant_Training Data Shape: (1084,)
Sushant_Test Data Shape: (272,)
Shruti_Training Data Shape: (1092,)
Shruti_Test Data Shape: (273,)
```

In [153]:

```
print(f"Number of Observations in Rishab: {dtv_rishab.shape[0]}\nTokens/Features in
print(f"Number of Observations in Sushant: {dtv_sushant.shape[0]}\nTokens/Features i
print(f"Number of Observations in Shruti: {dtv_shruti.shape[0]}\nTokens/Features in
```

```
Number of Observations in Rishab: 1116
Tokens/Features in Rishab: 521489
Number of Observations in Sushant: 1084
Tokens/Features in Sushant: 521489
Number of Observations in Shruti: 1092
Tokens/Features in Shruti: 521489
```

In [154]:

```
%%time
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC, SVC
from time import perf_counter
import warnings
warnings.filterwarnings(action='ignore')
models = {
    "Random Forest": {"model":RandomForestClassifier(), "perf":0},
    "MultinomialNB": {"model":MultinomialNB(), "perf":0},
    "KNN": {"model":KNeighborsClassifier(), "perf":0},
    "Decision Tree": {"model":DecisionTreeClassifier(), "perf":0},
    "SVM": {"model":LinearSVC(), "perf":0}
}

for name, model in models.items():
    start = perf_counter()
    model['model'].fit(dtv_rishab, y_train_rishab)
    duration = perf_counter() - start
    duration = round(duration,2)
    model["perf"] = duration
    print(f"{name:20} trained in {duration} sec in Rishab's dataset")
```

```
Random Forest        trained in 65.74 sec in Rishab's dataset
MultinomialNB        trained in 16.83 sec in Rishab's dataset
KNN                  trained in 87.83 sec in Rishab's dataset
Decision Tree        trained in 57.08 sec in Rishab's dataset
SVM                  trained in 5.17 sec in Rishab's dataset
Wall time: 4min 11s
```

In [158]:

```
for name, model in models.items():
    start = perf_counter()
    model['model'].fit(dtv_sushant, y_train_sushant)
    duration = perf_counter() - start
    duration = round(duration,2)
    model["perf"] = duration
    print(f"{name:20} trained in {duration} sec in Sushant's dataset")
```

```
Random Forest        trained in 104.44 sec in Sushant's dataset
MultinomialNB        trained in 10.31 sec in Sushant's dataset
KNN                  trained in 132.72 sec in Sushant's dataset
Decision Tree        trained in 134.36 sec in Sushant's dataset
SVM                  trained in 10.84 sec in Sushant's dataset
```

In [162]:

```
for name, model in models.items():
    start = perf_counter()
    model['model'].fit(dtv_shruti, y_train_shruti)
    duration = perf_counter() - start
    duration = round(duration,2)
    model["perf"] = duration
    print(f"{name:20} trained in {duration} sec in Shruti's dataset")
```

```
Random Forest        trained in 79.79 sec in Shruti's dataset
MultinomialNB        trained in 7.38 sec in Shruti's dataset
KNN                  trained in 110.01 sec in Shruti's dataset
Decision Tree        trained in 168.78 sec in Shruti's dataset
SVM                  trained in 11.67 sec in Shruti's dataset
```

In [155]:

```
test_dtv_rishab = cv.transform(X_test_rishab['Body'])
test_dtv_rishab = test_dtv_rishab.toarray()
print(f"Number of Observations in Rishab's test dataset: {test_dtv_rishab.shape[0]}\
```

```
Number of Observations in Rishab's test dataset: 280
Tokens: 521489
```

In [159]:

```
test_dtv_sushant = cv.transform(X_test_sushant['Body'])
test_dtv_sushant = test_dtv_sushant.toarray()
print(f"Number of Observations in Sushant's test dataset: {test_dtv_sushant.shape[0]
```

```
Number of Observations in Sushant's test dataset: 272
Tokens: 521489
```

In [163]:

```
test_dtv_shruti = cv.transform(X_test_shruti['Body'])
test_dtv_shruti = test_dtv_shruti.toarray()
print(f"Number of Observations in Shruti's test dataset: {test_dtv_shruti.shape[0]}\
```

```
Number of Observations in Shruti's test dataset: 273
Tokens: 521489
```

In [156]:

```
models_accuracy_rishab = []
for name, model in models.items():
    models_accuracy_rishab.append([name, model["model"].score(test_dtv_rishab, y_tes
```

In [160]:

```
models_accuracy_sushant = []
for name, model in models.items():
    models_accuracy_sushant.append([name, model["model"].score(test_dtv_sushant, y_t
```

In [164]:

```
models_accuracy_shruti = []
for name, model in models.items():
    models_accuracy_shruti.append([name, model["model"].score(test_dtv_shruti, y_tes
```

In [157]:

```
df_accuracy_rishab = pd.DataFrame(models_accuracy_rishab)
df_accuracy_rishab.columns = ['Model', 'Test Accuracy', 'Training time (sec)']
df_accuracy_rishab.sort_values(by = 'Test Accuracy', ascending = False, inplace=True
df_accuracy_rishab.reset_index(drop = True, inplace=True)
df_accuracy_rishab
```

Out[157]:

| | Model | Test Accuracy | Training time (sec) |
|---|---|---|---|
| 0 | Decision Tree | 0.892857 | 57.08 |
| 1 | SVM | 0.892857 | 5.17 |
| 2 | Random Forest | 0.875000 | 65.74 |
| 3 | MultinomialNB | 0.835714 | 16.83 |
| 4 | KNN | 0.832143 | 87.83 |

In [161]:

```
df_accuracy_sushant = pd.DataFrame(models_accuracy_sushant)
df_accuracy_sushant.columns = ['Model', 'Test Accuracy', 'Training time (sec)']
df_accuracy_sushant.sort_values(by = 'Test Accuracy', ascending = False, inplace=Tru
df_accuracy_sushant.reset_index(drop = True, inplace=True)
df_accuracy_sushant
```

Out[161]:

| | Model | Test Accuracy | Training time (sec) |
|---|---|---|---|
| 0 | Random Forest | 0.952206 | 104.44 |
| 1 | SVM | 0.933824 | 10.84 |
| 2 | Decision Tree | 0.882353 | 134.36 |
| 3 | MultinomialNB | 0.863971 | 10.31 |
| 4 | KNN | 0.856618 | 132.72 |

In [165]:

```
df_accuracy_shruti = pd.DataFrame(models_accuracy_shruti)
df_accuracy_shruti.columns = ['Model', 'Test Accuracy', 'Training time (sec)']
df_accuracy_shruti.sort_values(by = 'Test Accuracy', ascending = False, inplace=True
df_accuracy_shruti.reset_index(drop = True, inplace=True)
df_accuracy_shruti
```

Out[165]:

| | Model | Test Accuracy | Training time (sec) |
|---|---|---|---|
| 0 | SVM | 0.937729 | 11.67 |
| 1 | Random Forest | 0.919414 | 79.79 |
| 2 | MultinomialNB | 0.879121 | 7.38 |
| 3 | KNN | 0.879121 | 110.01 |
| 4 | Decision Tree | 0.871795 | 168.78 |

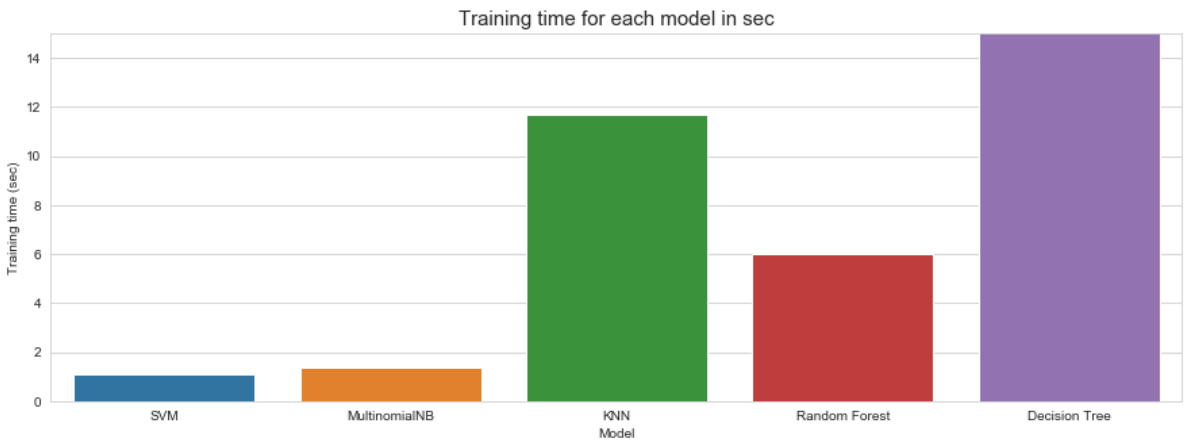
In [1]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

pal = sns.color_palette("Greens_d", len(df_accuracy))
rank = df_accuracy['Test Accuracy'].argsort().argsort()

plt.figure(figsize = (15,5))
g = sns.barplot(x = 'Model', y ='Test Accuracy', data = df_accuracy, palette=np.arra
plt.title('Accuracy on the test set\n', fontsize = 15)
plt.ylim(0.8,0.87)
for index, row in df_accuracy.iterrows():
    g.text(row.name,row['Test Accuracy'], round(row['Test Accuracy'],4), color='blac

plt.show()
```

...

In [61]:

```
pal = sns.color_palette("Greens_d", len(df_accuracy))
rank = df_accuracy['Training time (sec)'].argsort().argsort()

plt.figure(figsize = (15,5))
sns.barplot(x = 'Model', y = 'Training time (sec)', data = df_accuracy)
plt.title('Training time for each model in sec', fontsize = 15)
plt.ylim(0,15)
plt.show()
```




In [3]:

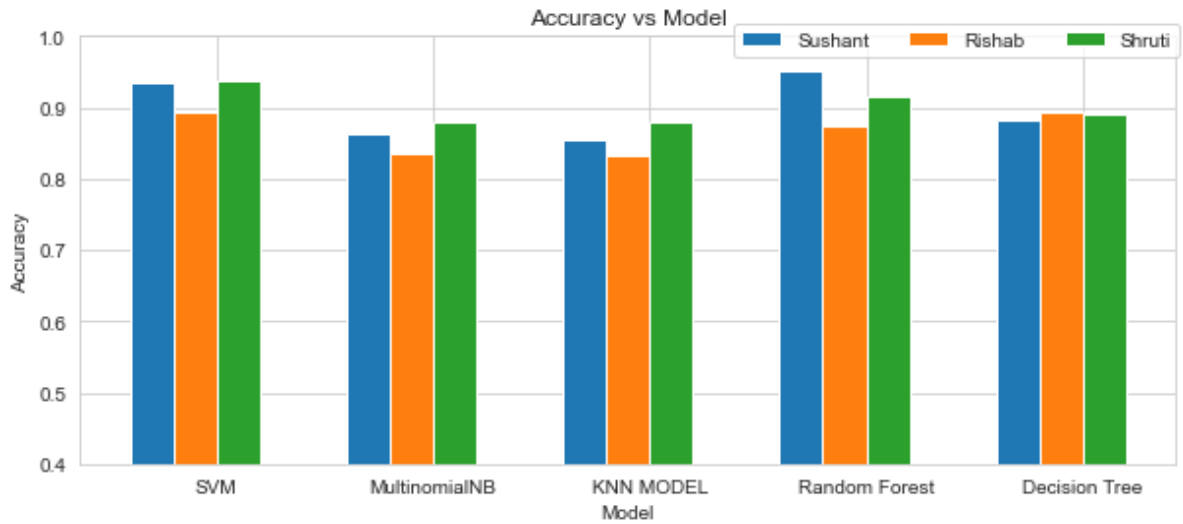
```
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:

```
```

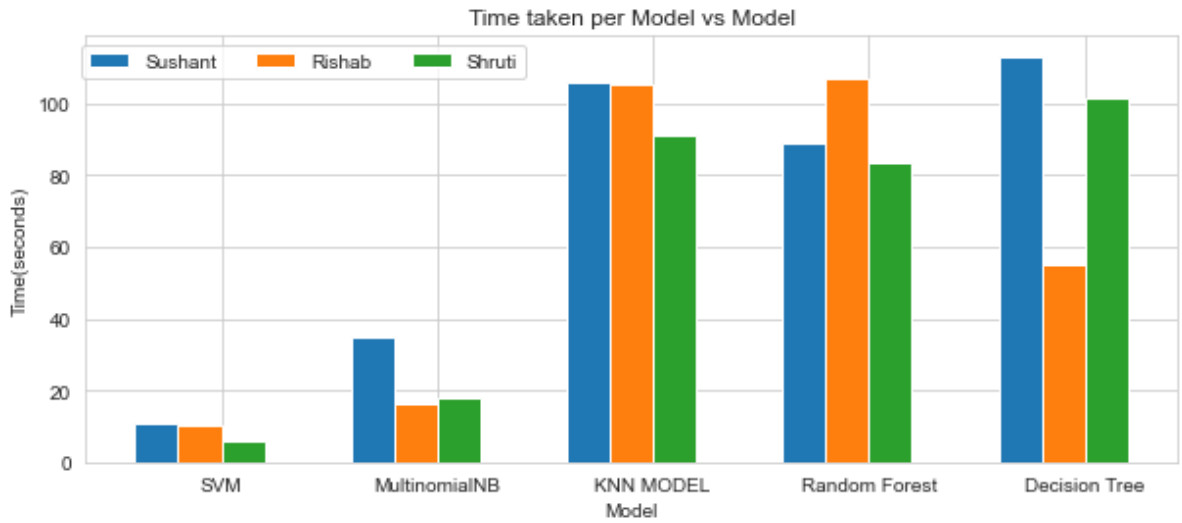
In [166]:

```
mods=['SVM','MultinomialNB','KNN MODEL','Random Forest','Decision Tree']
acc_rishab=[0.893,0.836,0.832,0.875,0.893]
acc_sushant=[0.934,0.864,0.856,0.952,0.882]
acc_shruti=[0.938,0.879,0.879,0.916,0.890]

X_axis = np.arange(len(mods))
plt.rcParams["figure.figsize"] = (10, 4)

plt.bar(X_axis - 0.3, acc_sushant, 0.2, label = 'Sushant')
plt.bar(X_axis - 0.1, acc_rishab, 0.2, label = 'Rishab')
plt.bar(X_axis + 0.1, acc_shruti, 0.2, label = 'Shruti')

plt.xticks(X_axis, mods)
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.title("Accuracy vs Model",loc='center')
plt.legend(loc="upper center", bbox_to_anchor=(0.8, 1.05), ncol=3)
plt.ylim(0.4,1)

plt.show()
```



In [84]:

```
mods=['SVM','MultinomialNB','KNN MODEL','Random Forest','Decision Tree']
time_rishab=[10.38,16.12,105.03,107.06,55.12]
time_sushant=[10.84,34.62,106.07,89.01,113.06]
time_shruti=[5.91,17.79,90.88,83.44,101.58]

X_axis = np.arange(len(mods))
plt.rcParams["figure.figsize"] = (10, 4)

plt.bar(X_axis - 0.3, time_sushant, 0.2, label = 'Sushant')
plt.bar(X_axis - 0.1, time_rishab, 0.2, label = 'Rishab')
plt.bar(X_axis + 0.1, time_shruti, 0.2, label = 'Shruti')

plt.xticks(X_axis, mods)
plt.xlabel("Model")
plt.ylabel("Time(seconds)")
plt.title("Time taken per Model vs Model",loc='center')
plt.legend(loc="upper center", bbox_to_anchor=(0.2, 1), ncol=3)
plt.show()
```




In [167]:

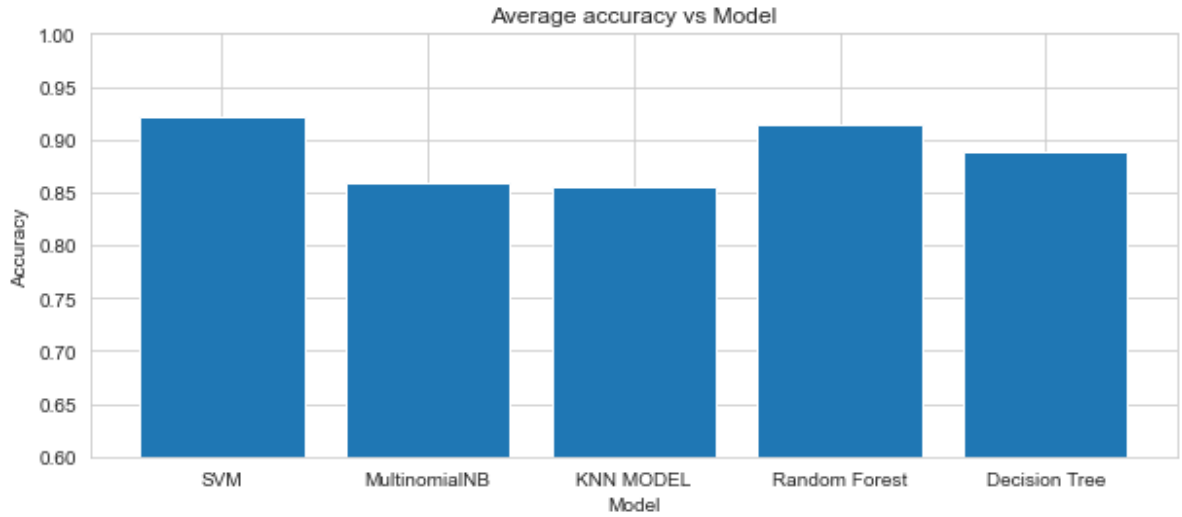
```
np_a_rishab=np.array(acc_rishab)
np_a_sushant=np.array(acc_sushant)
np_a_shruti=np.array(acc_shruti)
np_av_acc=np_a_rishab+np_a_sushant+np_a_shruti
np_av_acc=np_av_acc/3
np_av_acc
```

Out[167]:

```
array([0.92166667, 0.85966667, 0.85566667, 0.91433333, 0.88833333])
```

In [168]:

```
X_axis = np.arange(len(mods))
plt.rcParams["figure.figsize"] = (10, 4)

plt.bar(X_axis, np_av_acc)

plt.xticks(X_axis, mods)
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.title("Average accuracy vs Model",loc='center')
#plt.legend(loc="upper center", bbox_to_anchor=(0.2, 1), ncol=3)

plt.ylim(0.6,1)

plt.show()
```




####