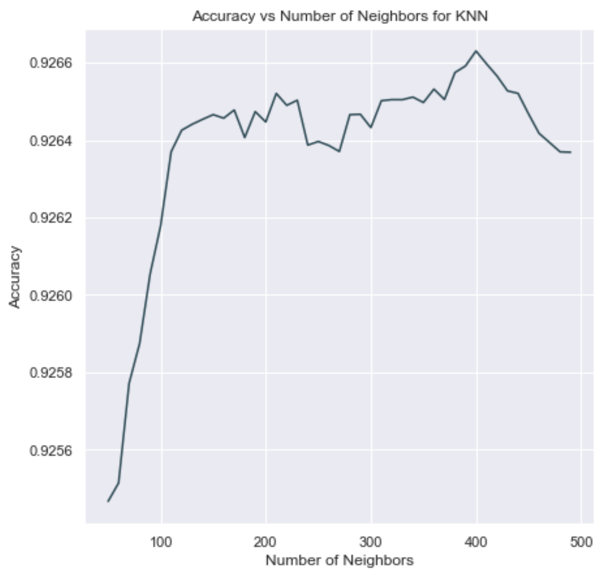
Model Tuning: K-Nearest Neighbor Regression

In our KNN model, we guessed a value of neighbors to be 60. However, we want to find a optimal number of neighbors to improve our model. We can do that by doing a search on a the neighbors value. Here is a graph of the accuracy and number of neighbors. I tested the number of neighbors from 50 to 500 in steps of 10. The optimal number of neighbors appear to be 400.



With 400, we got a model score of 92.663%. This is about 0.063% better than with 60 neighbors. A small but reasonable improvement.

```
model2 = KNeighborsRegressor(n_neighbors=400)
```

```
model2.fit(X_train, y_train)
```

```
KNeighborsRegressor(n_neighbors=400)
```

```
accuracy = model2.score(X_test,y_test)
accuracy
```

```
0.9266302971856202
```

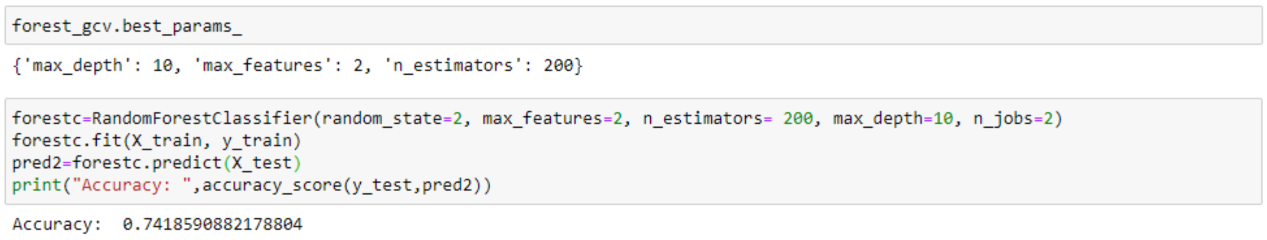
Model Tuning: Random Forest

We can also use GridSearchCV to tune the hyperparameters for the random forest. I ran the gridSearchCV with 5 fold CV.

```
param_grid = {
    'n_estimators': [200, 500],
    'max_features': [1,2,3,4,5],
    'max_depth' : [5,10,15,20,25,30,35,40,45,50]
}
```

```
from sklearn.model_selection import GridSearchCV

forest_gcv = GridSearchCV(estimator=forest, param_grid=param_grid, cv=5)
forest_gcv.fit(X_train, y_train)
```



```
forest_gcv.best_params_
```

```
{'max_depth': 10, 'max_features': 2, 'n_estimators': 200}
```

```
forestc=RandomForestClassifier(random_state=2, max_features=2, n_estimators= 200, max_depth=10, n_jobs=2)
forestc.fit(X_train, y_train)
pred2=forestc.predict(X_test)
print("Accuracy: ",accuracy_score(y_test,pred2))
```

```
Accuracy:  0.7418590882178804
```

However, with the random forest, I didn't really see an improvement after parameter tuning. The grid search already look over 45 minutes to run on my computer. Perhaps with a more robust computer and more values for searching it might be possible to get a better accuracy. However, practically speaking the parameters previously are sufficient. It appears that random forest performs roughly 74-75% in classifying whether an app will be popular or not. I think the lesson here is sometimes it might not be possible to see an improvement with tuning.

Further Work

There's definitely a lot of other things that could be done with the data set. There is a also a user reviews CSV, so it's possible to do some sentiment analysis and natural language processing to find how users feel about apps through their reviews. Another example would be classifying whether a review is fake or not. This data set from Kaggle is a bit old. I think it would be interesting to scrap current app data and analysis it. Especially during this covid time, certain apps have skyrocketed in popularity. It would be interesting to see if it's possible to build a model that can cluster which apps grew the most and which declined the most during the pandemic. Lastly, we can also try building a model that determines the price of an app. Pricing competitively is important to companies and many of companies even have pricing analysts. Overall, this project was fun and there's certainly a lot of further exploration that can be done. With a more powerful computer, it's possible to analysis a bigger data set of apps.