Practicum

FORECAST AND PREDICTION

PROJECT SUMMARY

- **Background**
  - *One of the most common problems gyms and other services face is customer churn. How do you know if a customer is no longer with you?*
- **Project Goal:**
  - *In order to fight churn, Model Fitness has digitized a number of its customer profiles. Your task is to analyze them and come up with a customer retention strategy.*
    - *Learn to predict the probability of churn (for the upcoming month) for each customer*
    - *Draw up typical user portraits: select the most outstanding groups and describe their main features*
    - *Analyze the factors that impact churn most*
    - *Draw basic conclusions and develop recommendations on how to improve customer service:*
    - *Identify target groups.*

PROJECT SUMMARY

- **Data**
    - *Model Fitness provided you with CSV files containing data on churn for a given month and information on the month preceding it.*
- **Steps:**
    - *EDA*
    - *Build a Model to predict user Churn*
    - *Create a user cluster*
    - *Come up with conclusions.*

Practicum

STEP 2 - EDA AND ANALYSIS

1. **Split the Data in two DF.**
   - **Left**
   - **Stayed**

Now I'll divide the data to 2 data frames (left and stayed) with only features by dropping churn

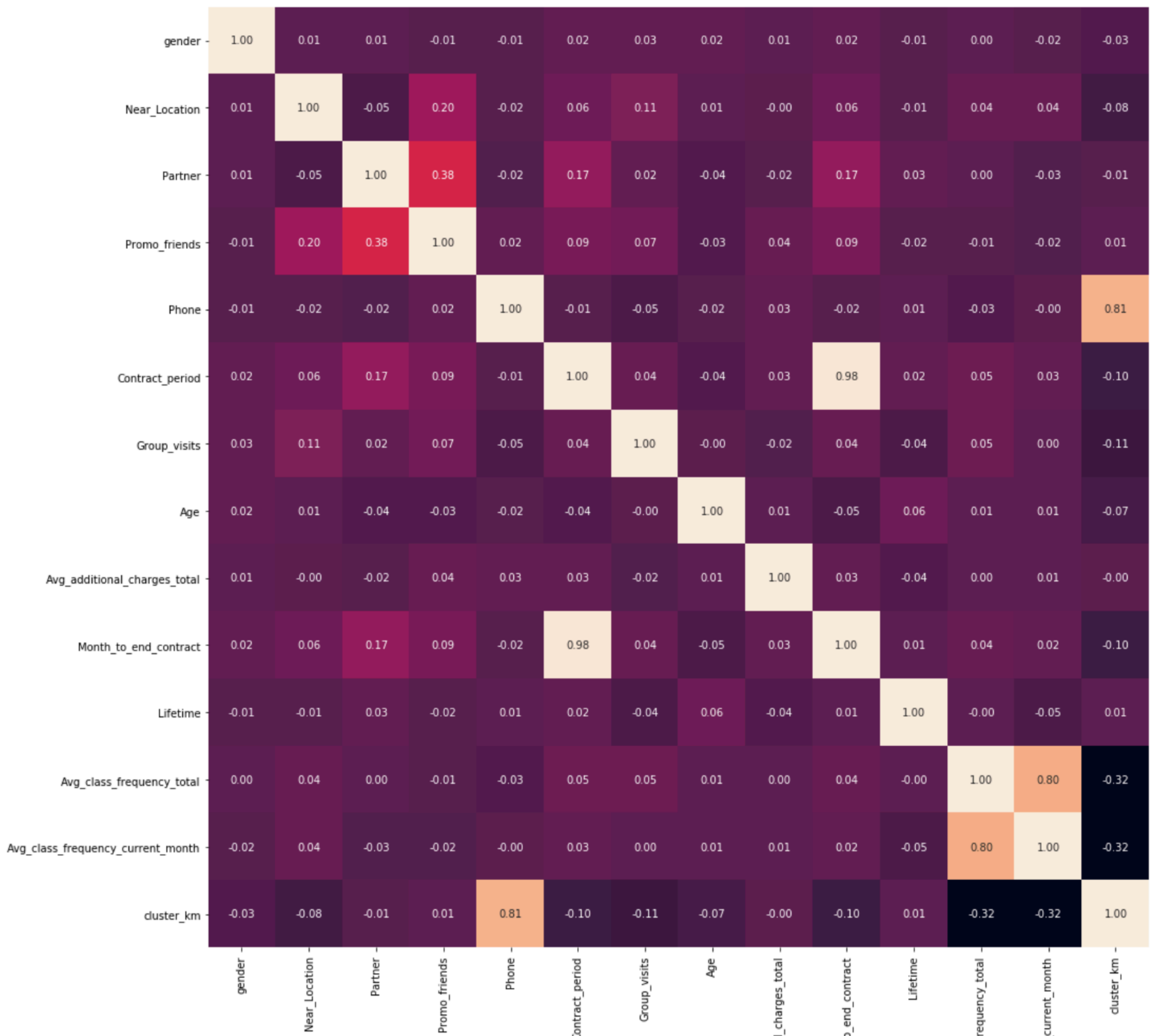
In [97]:
```
left = gym.query('Churn == 1')
left = left.drop('Churn', axis =1)
left

stayed = gym.query('Churn == 0')
stayed = stayed.drop('Churn', axis =1)
stayed
```

2. **Analyze each DF.**

3. **Correlation Matrix**

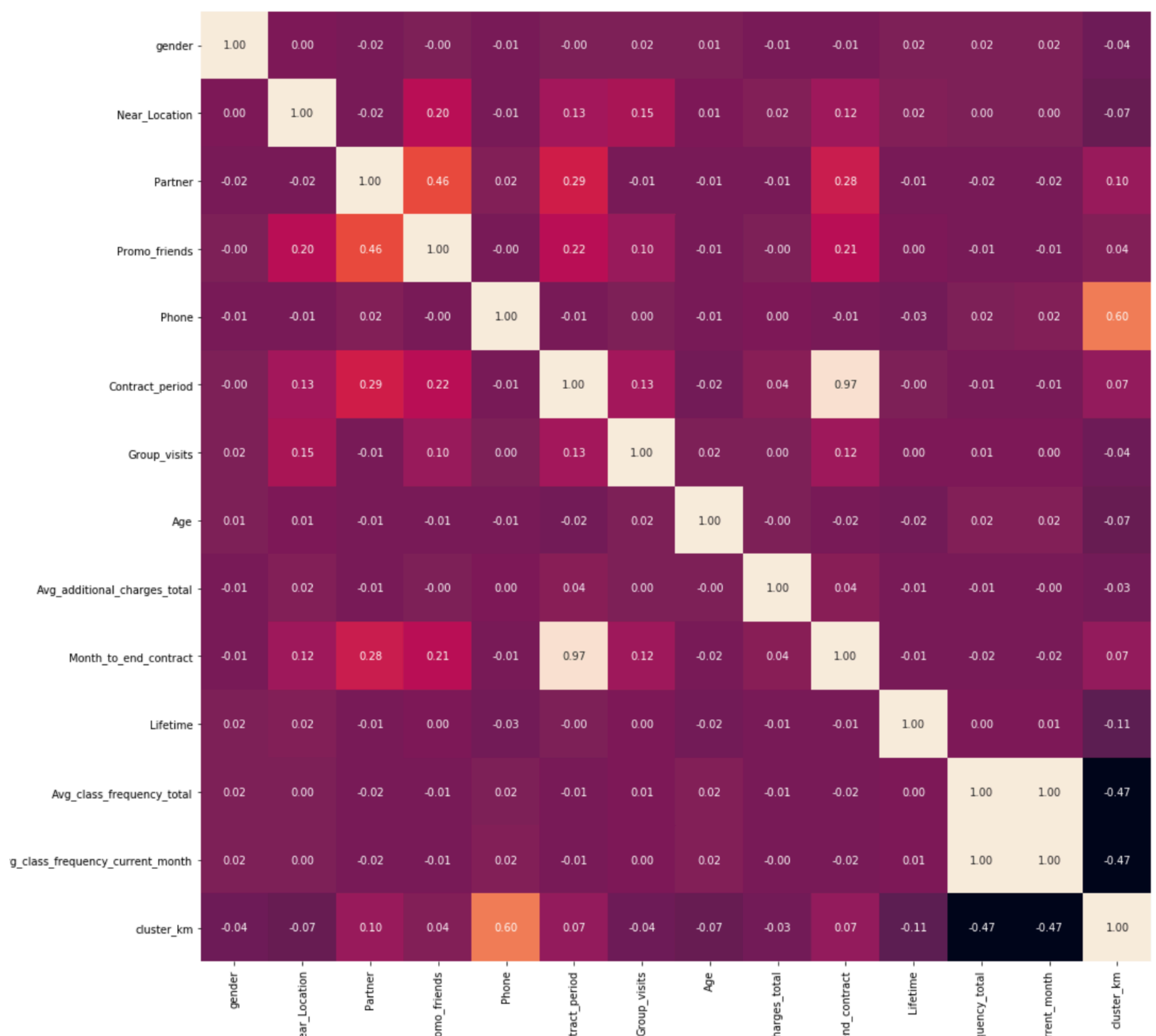
In [101]:
```
corr_l = left.corr()
plt.figure(figsize=(20, 20))
# plot a heatmap
ax = sb.heatmap(corr_l, annot=True,square = True , fmt=".2f")
```

STEP 2 - EDA AND ANALYSIS



STEP 2 - EDA AND ANALYSIS

In [102]:
```
corr_s = stayed.corr()
plt.figure(figsize=(20, 20))
# plot a heatmap
ax = sb.heatmap(corr_s, annot=True,square = True , fmt=".2f")
```



Practicum

STEP 3 - CREATE THE MODEL

- **Train the data**

```
# Dividing the data into features (the X matrix) and a target variable (y)
X = gym.drop('Churn', axis = 1)
y = gym['Churn']
# divide the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 0)
```

- **Use at least two models:**
  - **Logistic Regression**
  - **Random Forest**

```
# Logistic regression

# Creating an instance of the model
logicr_model = LogisticRegression(random_state=0)

# training the model on the training data set and storing the information learned from the data
logicr_model.fit(X_train, y_train)

# using the trained model to make forecasts
logicr_predictions = logicr_model.predict(X_test)
logicr_probabilities = logicr_model.predict_proba(X_test)[:,1]
```

Practicum

STEP 3 - CREATE THE MODEL

```
# Random forest

# defining the algorithm for the new random forest model
rforest_model = RandomForestClassifier(random_state=0)

# training the random forest model
rforest_model.fit(X_train, y_train)

# using the trained model to make predictions
rforest_predictions = rforest_model.predict(X_test)
rforest_probabilities = rforest_model.predict_proba(X_test)[:,1]
```

- **Accuracy Metrics to define which model is better**

```
def print_all_metrics(y_true, y_pred, y_proba, title = 'Classification metrics'):
    print(title)
    print('\tAccuracy: {:.2f}'.format(accuracy_score(y_true, y_pred)))
    print('\tPrecision: {:.2f}'.format(precision_score(y_true, y_pred)))
    print('\tRecall: {:.2f}'.format(recall_score(y_true, y_pred)))
```

```
Metrics for logistic regression:
        Accuracy: 0.91
        Precision: 0.82
        Recall: 0.80
```

```
print_all_metrics(y_test,rforest_predictions,rforest_probabilities, title = 'Metrics for random forest:')
```

```
Metrics for random forest:
        Accuracy: 0.92
        Precision: 0.84
        Recall: 0.82
```

It seems like random forest model is better at all metrics here and is much accurate for this case

STEP 3 - CREATE THE MODEL

```
# Random forest

# defining the algorithm for the new random forest model
rforest_model = RandomForestClassifier(random_state=0)

# training the random forest model
rforest_model.fit(X_train, y_train)

# using the trained model to make predictions
rforest_predictions = rforest_model.predict(X_test)
rforest_probabilities = rforest_model.predict_proba(X_test)[:,1]
```

- **Accuracy Metrics to define which model is better**

```
def print_all_metrics(y_true, y_pred, y_proba, title = 'Classification metrics'):
    print(title)
    print('\tAccuracy: {:.2f}'.format(accuracy_score(y_true, y_pred)))
    print('\tPrecision: {:.2f}'.format(precision_score(y_true, y_pred)))
    print('\tRecall: {:.2f}'.format(recall_score(y_true, y_pred)))
```

```
Metrics for logistic regression:
	Accuracy: 0.91
	Precision: 0.82
	Recall: 0.80
```

```
print_all_metrics(y_test,rforest_predictions,rforest_probabilities, title = 'Metrics for random forest:')
```

```
Metrics for random forest:
	Accuracy: 0.92
	Precision: 0.84
	Recall: 0.82
```

It seems like random forest model is better at all metrics here and is much accurate for this case

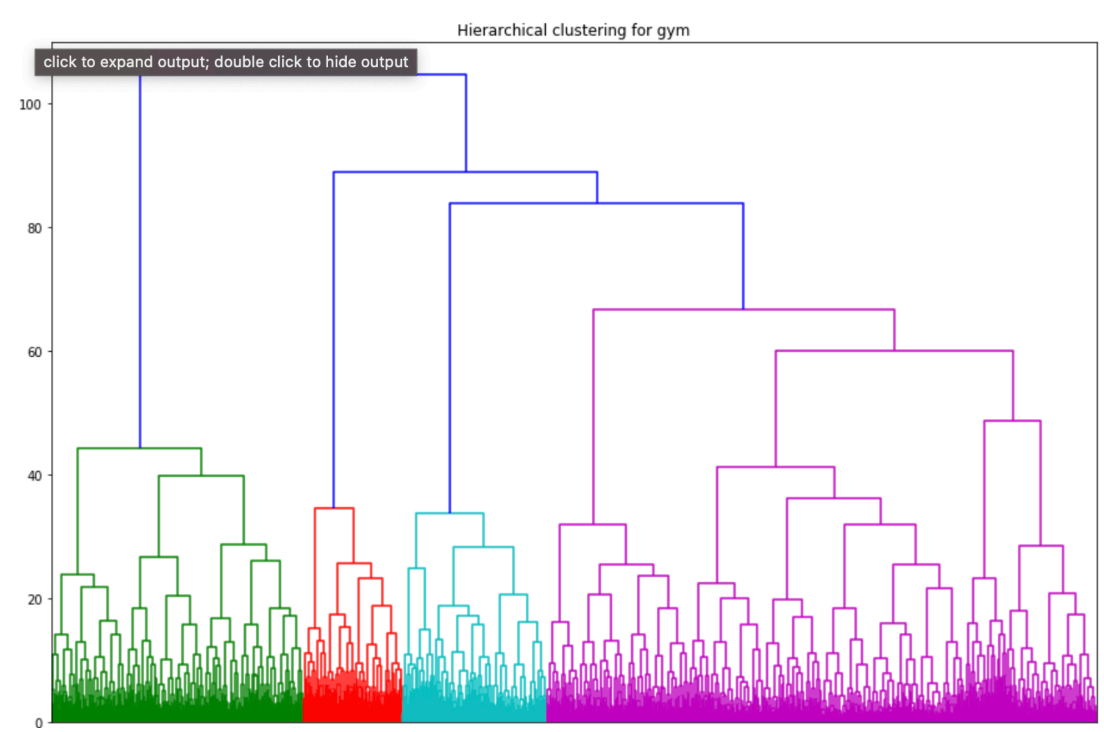
STEP 4 - CREATE A CLUSTER

- **First. Standarize the data**

```
scaler = StandardScaler() # creating a scaler class object (normalizer)
x_sc = scaler.fit_transform(X) # training the normalizer and transforming the dataset
linked = linkage(x_sc, method = 'ward')
```

- **Plot a dendrogram that represents the distance between clusters which will help me determine what is the number of clusters that is needed**

```
plt.figure(figsize=(15, 10))
dendrogram(linked, orientation='top')
plt.title('Hierarchical clustering for gym')
plt.show()
```



Practicum

STEP 4 - CREATE A CLUSTER

- **Conclusion:**
  - **Number of Clusters = 4**
- **Create K-mean model with n=4**

```
n = 4
#n = 5
km = KMeans(n_clusters = n , random_state = 0) # setting the number of clusters as 5
labels = km.fit_predict(x_sc) # applying the algorithm to the data and forming a cluster vector
```

I'll now add the labels that were created to the original data and I'll show average of everey feature per cluster (0-3)

```
gym['cluster_km'] = labels
gym.groupby(['cluster_km']).mean()
```

cluster_km	gender	Near_Location	Partner	Promo_friends	Phone	Contract_period	Group_visits	Age	Avg_additional_charges_total	Month_to_end_con
0	0.523316	0.862694	0.471503	0.305699	0.0	4.777202	0.427461	29.297927	144.208179	4.46
1	0.541588	0.865784	0.335539	0.199433	1.0	2.386578	0.450851	30.005671	157.889886	2.22
2	0.503697	0.940850	0.778189	0.573937	1.0	10.685767	0.533272	29.896488	161.102734	9.75
3	0.489145	0.755767	0.385346	0.192673	1.0	1.895522	0.291723	28.042062	129.409699	1.80

Q&A

THANK YOU!

Practicum