IDS Assignment - Android Malware Detection

Importing the required packages

In [2]:
```
import os
import math
import joblib
import warnings
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_scor
warnings.filterwarnings("ignore")
from IPython.core.display import HTML
```

Loading the dataset

We read from the csv and load it into a variable in order to perform exploratory data analysis and train a model

In [3]:
```
class Dataset:

    def __init__(self, data_path):
        self.data_path = data_path
        self.data = None

    def load_data(self):
        csv_data = pd.read_csv(self.data_path, sep=";")
        self.data = csv_data
        return self.data
```

Performing Exploratory Analysis on the data

Here we have defined functions to

1. Check for missing values
2. Perform principal component analysis (PCA) to check the spread of the dependent variable type between 2 components
3. Check for most permissions given by type
4. Check whether the dataset is balanced or imbalanced
5. Check the correlation between different columns
6. Remove columns which are correlated above a configurable threshold

In [61]:
```
class ExploratoryAnalysis:

    def __init__(self):
        self.data = Dataset("Dataset.csv").load_data()
        self.X = data.loc[:, data.columns != "type"]
        self.y = data.loc[:, data.columns == "type"]

    def analyse_missing_values(self):
        print("Missing values in the dataset is", self.data.isna().sum().sum())

    def perform_pca(self):
        pca_android_malware = PCA(2)  # project from 64 to 2 dimensions
        projected = pca_android_malware.fit_transform(self.X)
        print("Variance between the components", pca_android_malware.explained_var
        plt.scatter(projected[:, 0], projected[:, 1],
                    c=self.data.type, edgecolor='none', alpha=0.5,
                    cmap=plt.cm.get_cmap('Spectral', 10))
        plt.xlabel('component 1')
        plt.ylabel('component 2')
        plt.colorbar();

    def most_permissions_given(self):
        fig, axs =  plt.subplots(nrows=1)
        print("Most permissions given where type is Malicious")
        print(pd.Series.sort_values(self.data[self.data.type==1].sum(axis=0), asce
        pd.Series.sort_values(self.data[self.data.type==1].sum(axis=0), ascending=
        figs, axss =  plt.subplots(nrows=1)
        print("Most permissions given where type is Benign")
        print(pd.Series.sort_values(self.data[self.data.type==0].sum(axis=0), asce
        pd.Series.sort_values(self.data[self.data.type==0].sum(axis=0), ascending=
        plt.show()

    def distribution_type(self):
        fig, axs =  plt.subplots(nrows=1, sharex=True)
        self.data.type.value_counts().plot.bar(ax=axs, title="Distribution between
        plt.show()

    def correlation(self):
        corr_matrix = self.data.corr()
        f, ax = plt.subplots(figsize=(10, 8))
        dataplot = sns.heatmap(corr_matrix)
        plt.gca().axes.get_xaxis().set_visible(False)
        plt.gca().axes.get_yaxis().set_visible(False)
        plt.show()

    def remove_correlated_columns(self, threshold):
        col_corr = set()
        corr_matrix = self.data.corr()
        for i in range(len(corr_matrix.columns)):
            for j in range(i):
                if (corr_matrix.iloc[i, j] >= threshold) and (corr_matrix.columns[
                    colname = corr_matrix.columns[i]
                    if colname != "type":
                        col_corr.add(colname)
                        if colname in self.data.columns:
                            del self.data[colname]
```

```
        print("Number of columns with greater than " + str(threshold) + " correlat
        print("Columns with greater than " + str(threshold) + " correlation are",
        return self.data
```

Analysing data for missing values

Here we can see that the missing values across the complete dataset is 0.

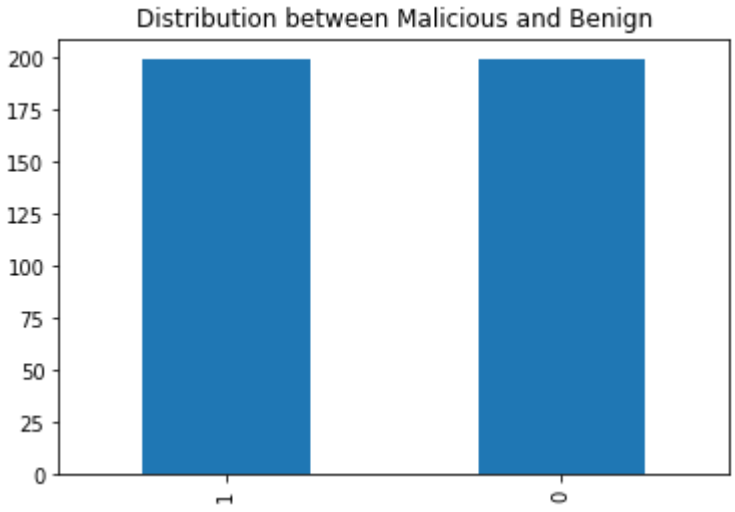
In [62]:
```
eda = ExploratoryAnalysis()
eda.analyse_missing_values()
```

```
Missing values in the dataset is 0
```

Checking whether the dataset is balanced or imbalanced

Based on the distribution of the dependent variable, we can see that the dataset is perfectly balanced.

In [25]:
```
eda.distribution_type()
```

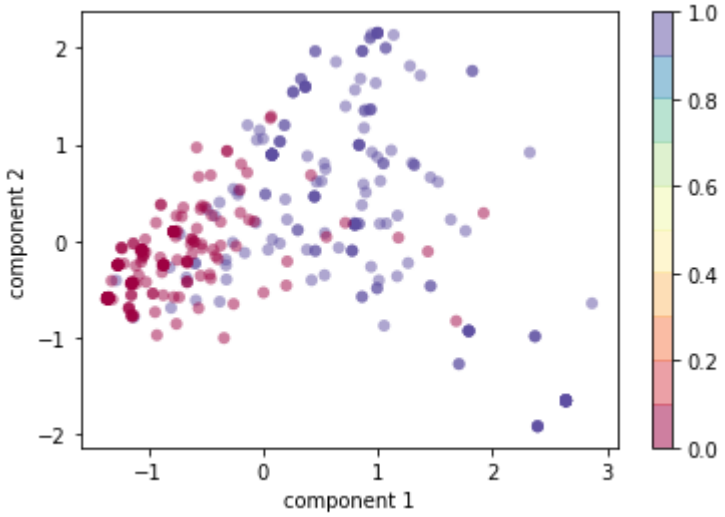


Performing PCA

When we plot the 2 components along with the different labels in dependent variable, can observe the 2 classes benign and malicious when projected to a 2-D space are separable to a certain extent.

In [63]:
```
eda.perform_pca()
```

```
Variance between the components [1.40738585 0.7913164 ]
```

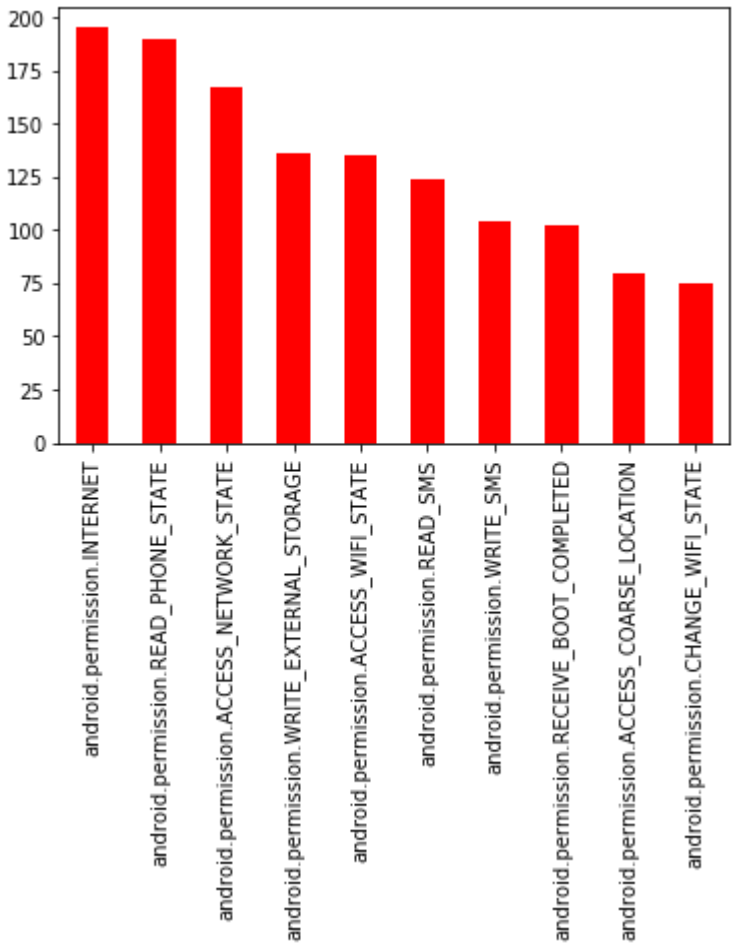


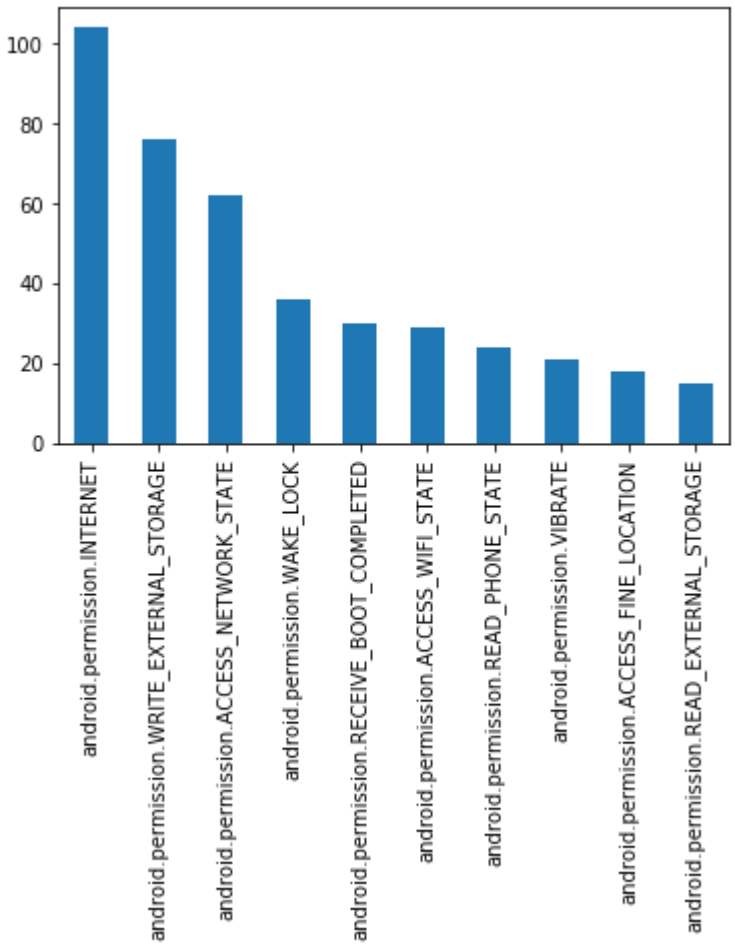
Top 10 permissions given for each type

We have plotted the top 10 permissions for each label in dependent variable. We can see that there is a 50% overlap in the top 10 categories for each type.

In [8]:
```
eda.most_permissions_given()
```

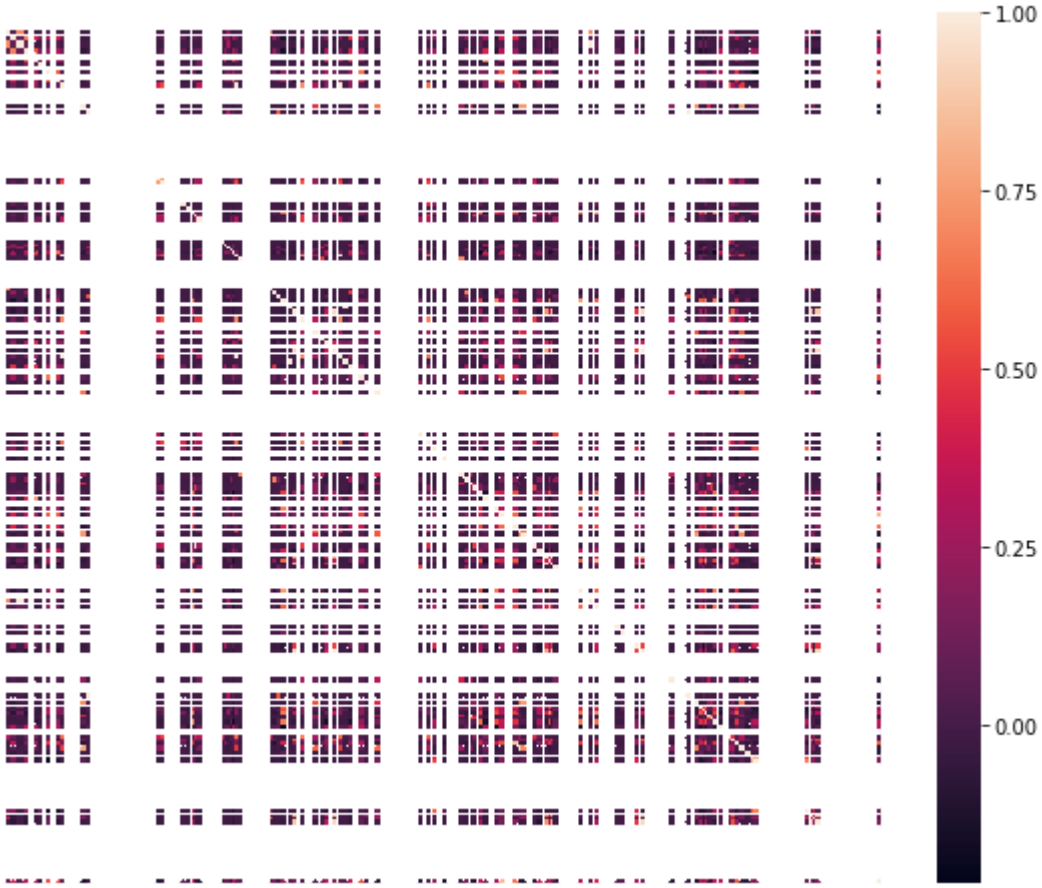
```
Most permissions given where type is Malicious
android.permission.INTERNET                   195
android.permission.READ_PHONE_STATE           190
android.permission.ACCESS_NETWORK_STATE       167
android.permission.WRITE_EXTERNAL_STORAGE     136
android.permission.ACCESS_WIFI_STATE          135
android.permission.READ_SMS                   124
android.permission.WRITE_SMS                  104
android.permission.RECEIVE_BOOT_COMPLETED     102
android.permission.ACCESS_COARSE_LOCATION      80
android.permission.CHANGE_WIFI_STATE           75
dtype: int64
Most permissions given where type is Benign
android.permission.INTERNET                   104
android.permission.WRITE_EXTERNAL_STORAGE      76
android.permission.ACCESS_NETWORK_STATE        62
android.permission.WAKE_LOCK                   36
android.permission.RECEIVE_BOOT_COMPLETED      30
android.permission.ACCESS_WIFI_STATE           29
android.permission.READ_PHONE_STATE            24
android.permission.VIBRATE                     21
android.permission.ACCESS_FINE_LOCATION        18
android.permission.READ_EXTERNAL_STORAGE       15
dtype: int64
```




100
80
60
40
20
0
android.permission.INTERNET
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.ACCESS_NETWORK_STATE
android.permission.WAKE_LOCK
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.ACCESS_WIFI_STATE
android.permission.READ_PHONE_STATE
android.permission.VIBRATE
android.permission.ACCESS_FINE_LOCATION
android.permission.READ_EXTERNAL_STORAGE

Performing correlation check between all columns

In [9]:
```
eda.correlation()
```



Importing required packages for Feature Engineering

In [30]:
```
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.ensemble import ExtraTreesClassifier
```

Feature Engineering

Here we have implemented the 2 feature engineering approaches. The first technique removes correlated columns above 0.4 threshold and then performs sklearn feature selection to select the top features. The second technique removes correlated columns above 0.6 threshold and then performs ExtraTreesClassifier to select the important features.

In [28]:
```
class FeatureEngineering:

    def __init__(self):
        self.data = Dataset("Dataset.csv").load_data()
        self.X = self.data.loc[:, self.data.columns != "type"]
        self.y = self.data.loc[:, self.data.columns == "type"]

    def feature_engineering_one(self):
        eda = ExploratoryAnalysis()
        eda.remove_correlated_columns(0.4)
        best_features = SelectKBest(score_func=chi2)
        fit = best_features.fit(self.X, self.y)
        df_scores = pd.DataFrame(fit.scores_)
        df_columns = pd.DataFrame(self.X.columns)
        feature_scores = pd.concat([df_columns, df_scores], axis=1)
        feature_scores.columns = ["Specs", "Scores"]
        print("\n Top 10 important features are \n", feature_scores.nlargest(10, '
        ax = feature_scores.nlargest(10, 'Scores').plot(kind="barh")
        ax.set_yticklabels(feature_scores.nlargest(10, 'Scores')["Specs"], rotatio
        for feature_column in feature_scores.iterrows():
            if feature_column[1]["Scores"] == 0 or math.isnan(feature_column[1]["S
                del self.data[feature_column[1]["Specs"]]
        return self.data

    def feature_engineering_two(self):
        eda = ExploratoryAnalysis()
        eda.remove_correlated_columns(0.6)
        model = ExtraTreesClassifier()
        model.fit(self.X, self.y)
        feat_importances = pd.Series(model.feature_importances_, index=self.X.colu
        print("\n Top 10 important features are \n", feat_importances.nlargest(10)
        feat_importances.nlargest(10).plot(kind="barh")
        for feature, feature_score in feat_importances.items():
            if feature_score == 0:
                del self.data[feature]
        return self.data
```

Analysing the results of Feature Engineering 1

In the first technique, we see that 57 columns having greater than 0.4 correlation are removed. And there are 88 columns left from initial 330 columns after selecting the top important features.

In [31]:
```
fe = FeatureEngineering()
fe.feature_engineering_one()
```

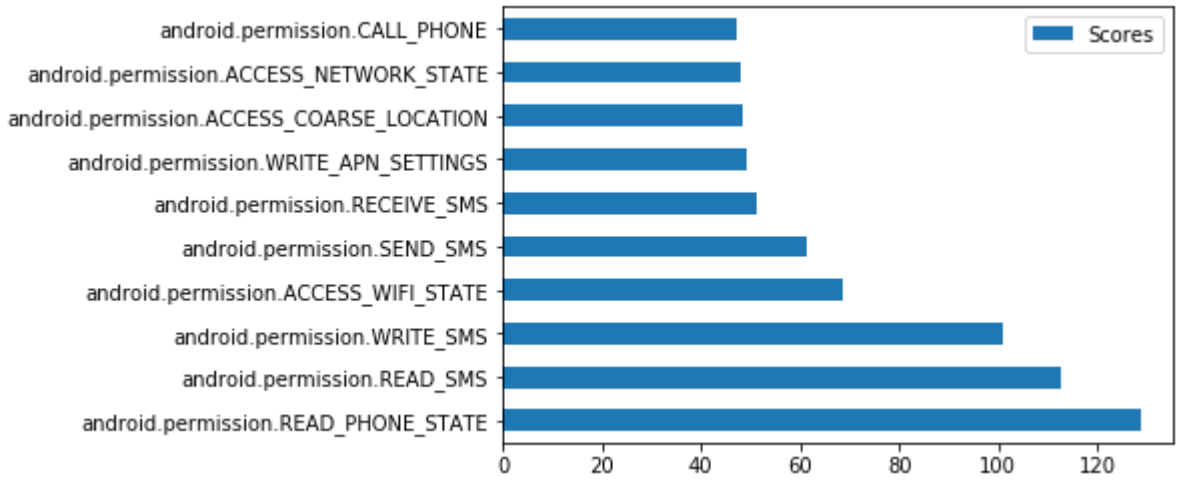
```
Number of columns with greater than 0.4 correlation are 57
Columns with greater than 0.4 correlation are {'android.permission.DEVICE_POWE
R', 'android.permission.RECORD_AUDIO', 'android.permission.BLUETOOTH_ADMIN',
'android.permission.FLASHLIGHT', 'android.permission.WRITE_SYNC_SETTINGS', 'an
droid.permission.SEND_SMS', 'com.android.launcher.permission.INSTALL_SHORTCU
T', 'android.permission.READ_LOGS', 'android.permission.MODIFY_PHONE_STATE',
'android.permission.READ_SYNC_SETTINGS', 'android.permission.RECEIVE_MMS', 'co
m.android.launcher.permission.UNINSTALL_SHORTCUT', 'android.permission.RESTART
_PACKAGES', 'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.US
E_CREDENTIALS', 'android.permission.READ_CALENDAR', 'android.permission.ACCESS
_DOWNLOAD_MANAGER', 'android.permission.READ_SYNC_STATS', 'android.permission.
ACCESS_DOWNLOAD_MANAGER_ADVANCED', 'android.permission.INTERNET', 'android.per
mission.GET_PACKAGE_SIZE', 'android.permission.WRITE_APN_SETTINGS', 'android.p
ermission.RECEIVE_WAP_PUSH', 'com.android.launcher.permission.WRITE_SETTINGS',
'android.permission.WRITE_CALENDAR', 'android.permission.GET_ACCOUNTS', 'andro
id.permission.DISABLE_KEYGUARD', 'android.permission.READ_SMS', 'android.permi
ssion.ACCESS_FINE_LOCATION', 'android.permission.BLUETOOTH', 'android.permissi
on.ACCESS_LOCATION_EXTRA_COMMANDS', 'android.permission.SET_WALLPAPER_HINTS',
'android.permission.WRITE_CONTACTS', 'android.permission.PROCESS_OUTGOING_CALL
S', 'android.permission.SET_WALLPAPER', 'android.permission.INJECT_EVENTS', 'a
ndroid.permission.GLOBAL_SEARCH_CONTROL', 'android.permission.RECEIVE_SMS', 'a
ndroid.permission.PERSISTENT_ACTIVITY', 'android.permission.SEND_DOWNLOAD_COMP
LETED_INTENTS', 'android.permission.READ_CONTACTS', 'android.permission.WRITE_
SMS', 'android.permission.HARDWARE_TEST', 'android.permission.UPDATE_DEVICE_ST
ATS', 'com.android.browser.permission.WRITE_HISTORY_BOOKMARKS', 'android.permi
ssion.WRITE_SECURE_SETTINGS', 'android.permission.READ_PHONE_STATE', 'android.
permission.INSTALL_PACKAGES', 'android.permission.CHANGE_WIMAX_STATE', 'androi
d.permission.READ_CALL_LOG', 'android.permission.WRITE_SETTINGS', 'android.per
mission.ACCESS_WIFI_STATE', 'android.permission.DELETE_CACHE_FILES', 'com.andr
oid.launcher.permission.READ_SETTINGS', 'android.permission.REBOOT', 'android.
permission.VIBRATE', 'android.permission.MANAGE_ACCOUNTS'}

 Top 10 important features are
                                          Specs      Scores
190        android.permission.READ_PHONE_STATE  128.766355
195                android.permission.READ_SMS  112.500000
277               android.permission.WRITE_SMS  101.038095
26        android.permission.ACCESS_WIFI_STATE   68.512195
225                android.permission.SEND_SMS   61.493151
208             android.permission.RECEIVE_SMS   51.428571
266      android.permission.WRITE_APN_SETTINGS   49.075472
9    android.permission.ACCESS_COARSE_LOCATION   48.268817
22     android.permission.ACCESS_NETWORK_STATE   48.144105
76                android.permission.CALL_PHONE   47.032258
```

Out[31]:

	android.permission.ACCESS_CACHE_FILESYSTEM	android.permission.ACCESS_COARSE_LOCATI
0	0	
1	0	
2	0	
3	0	

	android.permission.ACCESS_CACHE_FILESYSTEM	android.permission.ACCESS_COARSE_LOCATI
4	0	
...	...	
393	0	
394	0	
395	0	
396	0	
397	0	

398 rows × 88 columns



Analysing the results of Feature Engineering 2

In the first technique, we see that 41 columns having greater than 0.6 correlation are removed. And there are 88 columns left from initial 330 columns after selecting the top important features.

In [32]:
```
fe = FeatureEngineering()
fe.feature_engineering_two()
```

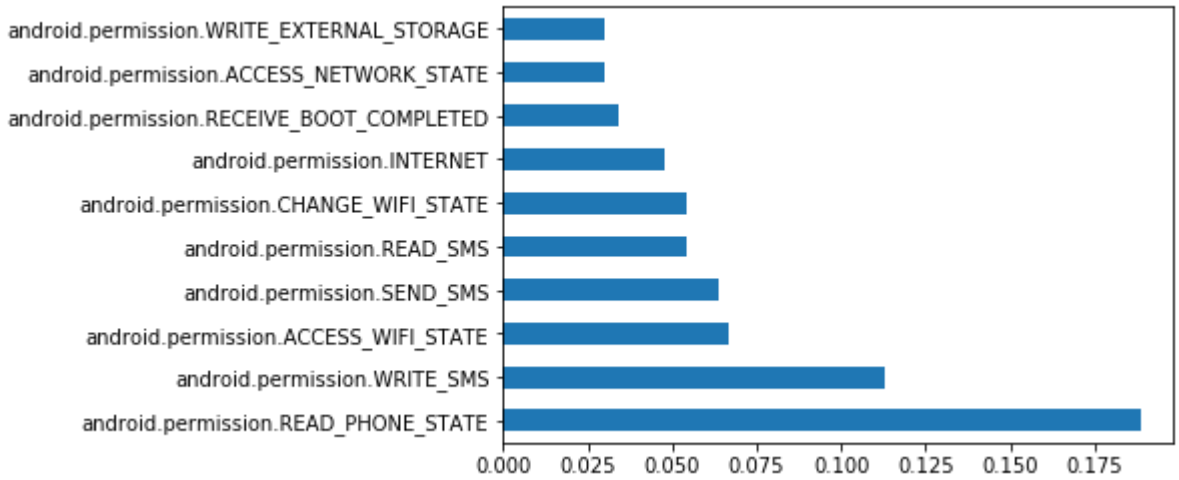
```
Number of columns with greater than 0.6 correlation are 41
Columns with greater than 0.6 correlation are {'android.permission.BLUETOOTH_A
DMIN', 'android.permission.FLASHLIGHT', 'android.permission.CHANGE_WIFI_STAT
E', 'android.permission.WRITE_SYNC_SETTINGS', 'android.permission.SEND_SMS',
'android.permission.MODIFY_PHONE_STATE', 'android.permission.READ_SYNC_SETTING
S', 'android.permission.RECEIVE_MMS', 'android.permission.RESTART_PACKAGES',
'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.USE_CREDENTIAL
S', 'android.permission.ACCESS_DOWNLOAD_MANAGER', 'android.permission.READ_SYN
C_STATS', 'android.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED', 'android.perm
ission.INTERNET', 'android.permission.WRITE_APN_SETTINGS', 'android.permissio
n.RECEIVE_WAP_PUSH', 'com.android.launcher.permission.WRITE_SETTINGS', 'androi
d.permission.WRITE_CALENDAR', 'android.permission.DISABLE_KEYGUARD', 'android.
permission.READ_SMS', 'android.permission.ACCESS_FINE_LOCATION', 'android.perm
ission.SET_WALLPAPER_HINTS', 'android.permission.WRITE_CONTACTS', 'android.per
mission.PROCESS_OUTGOING_CALLS', 'android.permission.INJECT_EVENTS', 'android.
permission.GLOBAL_SEARCH_CONTROL', 'android.permission.RECEIVE_SMS', 'android.
permission.PERSISTENT_ACTIVITY', 'android.permission.SEND_DOWNLOAD_COMPLETED_I
NTENTS', 'android.permission.READ_CONTACTS', 'android.permission.HARDWARE_TES
T', 'android.permission.UPDATE_DEVICE_STATS', 'com.android.browser.permission.
WRITE_HISTORY_BOOKMARKS', 'android.permission.WRITE_SECURE_SETTINGS', 'androi
d.permission.CHANGE_WIMAX_STATE', 'android.permission.WRITE_SETTINGS', 'androi
d.permission.DELETE_CACHE_FILES', 'com.android.launcher.permission.READ_SETTIN
GS', 'android.permission.REBOOT', 'android.permission.MANAGE_ACCOUNTS'}

 Top 10 important features are
 android.permission.READ_PHONE_STATE          0.188971
android.permission.WRITE_SMS                 0.112872
android.permission.ACCESS_WIFI_STATE         0.066499
android.permission.SEND_SMS                  0.063599
android.permission.READ_SMS                  0.054155
android.permission.CHANGE_WIFI_STATE         0.054131
android.permission.INTERNET                  0.047568
android.permission.RECEIVE_BOOT_COMPLETED    0.034207
android.permission.ACCESS_NETWORK_STATE      0.029912
android.permission.WRITE_EXTERNAL_STORAGE    0.029854
dtype: float64
```

Out[32]:

	android.permission.ACCESS_CACHE_FILESYSTEM	android.permission.ACCESS_COARSE_LOC
0	0	
1	0	
2	0	
3	0	
4	0	
...	...	
393	0	
394	0	
395	0	

	android.permission.ACCESS_CACHE_FILESYSTEM	android.permission.ACCESS_COARSE_LOC
396	0	
397	0	

398 rows × 88 columns



Importing packages to Train Model

In [35]:
```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import xgboost as xgb
```

In [36]:
```
estimator_list = {
    "logistic": "LogisticRegression",
    "decision_tree": "tree.DecisionTreeClassifier",
    "naive_bayes":"MultinomialNB",
    "random_forest":"RandomForestClassifier",
    "xgboost": "xgb.XGBClassifier",
    "svc": "SVC"
}
```

In [37]:
```
def get_estimators():
    return estimator_list.keys()
```

Building pipeline of algorithms

Building a pipeline of algorithms to be used later in the training and evaluation steps.

In [59]:
```
def load_pipeline(estimator="logistic"):
    if estimator == "logistic":
        pipeline = Pipeline([
            ('clf', LogisticRegression(random_state=19))
        ])
    elif estimator == "decision_tree":
        pipeline = Pipeline([
            ("clf", tree.DecisionTreeClassifier(max_depth=3, random_state=42))
        ])
    elif estimator == "naive_bayes":
        pipeline = Pipeline([
            ('clf', MultinomialNB())
        ])
    elif estimator == "random_forest":
        pipeline = Pipeline([
            ('clf', RandomForestClassifier(n_estimators=1000, oob_score=True, n_jo
                                           random_state=50, max_features="auto",
                                           max_leaf_nodes=30))
        ])
    elif estimator == "xgboost":
        pipeline = Pipeline([
            ('clf', xgb.XGBClassifier())
        ])
    elif estimator == "svc":
        pipeline = Pipeline([
            ('clf', SVC(kernel='linear'))
        ])
    else:
        print("estimator unavailable in pipeline")
    return pipeline
```

Building the train pipeline

Here we have built the functions to train, evaluate and save the model.

In [64]:
```
class Trainer:

    def __init__(self, train_test_split_ratio=0.2, estimator="logistic", model_nam
        self.train_test_split_ratio = train_test_split_ratio
        if choose_feat_eng is None:
            self.data = ExploratoryAnalysis().remove_correlated_columns(0.5)
        elif choose_feat_eng == "fe1":
            fe = FeatureEngineering()
            self.data = fe.feature_engineering_one()
        elif choose_feat_eng == "fe2":
            fe = FeatureEngineering()
            self.data = fe.feature_engineering_two()
        self.random_state = 101
        self.X = self.data.loc[:, self.data.columns != "type"]
        self.y = self.data.loc[:, self.data.columns == "type"]
        self.train_x, self.test_x, self.train_y, self.test_y = train_test_split(se
                                                                                 te
                                                                                 ra

        self.estimator = estimator
        self.model_name = model_name
        self.pipeline = None
        self.classification_report_test, self.classification_report_train, self.co
        self.choose_best_model = choose_best_model

    def train(self):
        """
            select the best algorithm from the different ones available and
            train the model
        """
        if self.choose_best_model is True:
            best_accuracy = 0
            best_estimator = self.estimator
            for estimator in get_estimators():
                self.estimator = estimator
                print("\n Evaluation for estimator ********** ", estimator)
                self.pipeline = load_pipeline(self.estimator)
                print(self.pipeline)
                self.pipeline.fit(self.train_x, self.train_y)
                accuracy = self.eval("accuracy")
                print("Accuracy is = ", accuracy)
                if best_accuracy < accuracy:
                    best_accuracy = accuracy
                    best_estimator = estimator

            print("Chosen best estimator = ", best_estimator, "best accuracy is =
            self.estimator = best_estimator
        self.pipeline = load_pipeline(self.estimator)
        self.pipeline.fit(self.train_x, self.train_y)

    def eval(self, metric=None):
        """
            get all the different evaluation metrics to measure the performance of
        """
        predictions_test = self.pipeline.predict(self.test_x)
        predictions_train = self.pipeline.predict(self.train_x)
        accuracy = accuracy_score(self.test_y, predictions_test)
```

```
        self.precision_score = precision_score(self.test_y, predictions_test, aver
        self.recall_score = recall_score(self.test_y, predictions_test, average='m
        self.classification_report_test = classification_report(self.test_y, predi
        self.classification_report_train = classification_report(self.train_y, pre
        self.confusion_matrix_test = confusion_matrix(self.test_y, predictions_tes
        self.confusion_matrix_train = confusion_matrix(self.train_y, predictions_t
        if metric == None:
            print("\n Accuracy is", accuracy)
            print("Precision", self.precision_score)
            print("Recall", self.recall_score)
            print("\n Train Classification Report \n", self.classification_report_
            print("\n Test Classification Report \n", self.classification_report_t
            print("\n Train Confusion Matrix \n", self.confusion_matrix_train)
            print("\n Test Confusion Matrix \n", self.confusion_matrix_test)
        elif metric == "accuracy":
            return accuracy_score(self.test_y, predictions_test)
        #return self.classification_report_test, self.classification_report_train,

    def save(self, path=os.getcwd()):
        """
            save the model after training
            :param path: the path where the model needs to be saved
        """
        path = os.path.join(path, "models")
        if not os.path.exists(path):
            os.mkdir(path)
        joblib.dump(self.pipeline, os.path.join(path, 'classifier_' + self.model_n
```

Performing initial evaluation of data with different algorithms

Initially running the data with no feature engineering and with 6 different algorithms to see which algorithms work best with the given data. The top 2 best algorithms will be selected for model training purposes.

The algorithms are:

1. Logistic Regression
2. Multinomial Naive Bayes
3. Decision Trees Classifier
4. Random Forest Classifier
5. XGBoost
6. SVC

Based on the evaluation run, we cann see that the algorithms have the accuracy metrics as follows.

1. Logistic Regression - 0.9125
2. Decision Trees Classifier - 0.9
3. Multinomial Naive Bayes - 0.825
4. Random Forest Classifier - 0.9125
5. XGBoost - 0.925

6. SVC - 0.8875

Hence we have chosen Random Forest Classifier as our ML1 and XGBoost as our ML2.

In [65]:
```
tr = Trainer(model_name="malware_detection_no_fe", choose_best_model=True, choose_
tr.train()
```

```
Number of columns with greater than 0.5 correlation are 48
Columns with greater than 0.5 correlation are {'android.permission.DEVICE_POWE
R', 'android.permission.RECORD_AUDIO', 'android.permission.BLUETOOTH_ADMIN',
'android.permission.FLASHLIGHT', 'android.permission.CHANGE_WIFI_STATE', 'andr
oid.permission.WRITE_SYNC_SETTINGS', 'android.permission.SEND_SMS', 'com.andro
id.launcher.permission.INSTALL_SHORTCUT', 'android.permission.MODIFY_PHONE_STA
TE', 'android.permission.READ_SYNC_SETTINGS', 'android.permission.RECEIVE_MM
S', 'android.permission.RESTART_PACKAGES', 'android.permission.READ_EXTERNAL_S
TORAGE', 'android.permission.USE_CREDENTIALS', 'android.permission.ACCESS_DOWN
LOAD_MANAGER', 'android.permission.READ_SYNC_STATS', 'android.permission.ACCES
S_DOWNLOAD_MANAGER_ADVANCED', 'android.permission.INTERNET', 'android.permissi
on.WRITE_APN_SETTINGS', 'android.permission.RECEIVE_WAP_PUSH', 'com.android.la
uncher.permission.WRITE_SETTINGS', 'android.permission.WRITE_CALENDAR', 'andro
id.permission.DISABLE_KEYGUARD', 'android.permission.READ_SMS', 'android.permi
ssion.ACCESS_FINE_LOCATION', 'android.permission.SET_WALLPAPER_HINTS', 'androi
d.permission.WRITE_CONTACTS', 'android.permission.PROCESS_OUTGOING_CALLS', 'an
droid.permission.INJECT_EVENTS', 'android.permission.GLOBAL_SEARCH_CONTROL',
'android.permission.RECEIVE_SMS', 'android.permission.PERSISTENT_ACTIVITY', 'a
ndroid.permission.SEND_DOWNLOAD_COMPLETED_INTENTS', 'android.permission.READ_C
ONTACTS', 'android.permission.HARDWARE_TEST', 'android.permission.UPDATE_DEVIC
E_STATS', 'com.android.browser.permission.WRITE_HISTORY_BOOKMARKS', 'android.p
ermission.WRITE_SECURE_SETTINGS', 'android.permission.READ_PHONE_STATE', 'andr
oid.permission.WRITE_CALL_LOG', 'android.permission.CHANGE_WIMAX_STATE', 'andr
oid.permission.WRITE_SETTINGS', 'android.permission.ACCESS_WIFI_STATE', 'andro
id.permission.DELETE_CACHE_FILES', 'com.android.launcher.permission.READ_SETTI
NGS', 'android.permission.REBOOT', 'android.permission.VIBRATE', 'android.perm
ission.MANAGE_ACCOUNTS'}

 Evaluation for estimator ***********  logistic
Pipeline(memory=None,
     steps=[('clf', LogisticRegression(C=1.0, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=19, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False))])
Accuracy is =  0.9125

 Evaluation for estimator ***********  decision_tree
Pipeline(memory=None,
     steps=[('clf', DecisionTreeClassifier(class_weight=None, criterion='gin
i', max_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=42,
            splitter='best'))])
Accuracy is =  0.9

 Evaluation for estimator ***********  naive_bayes
Pipeline(memory=None,
     steps=[('clf', MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e))])
Accuracy is =  0.825
```

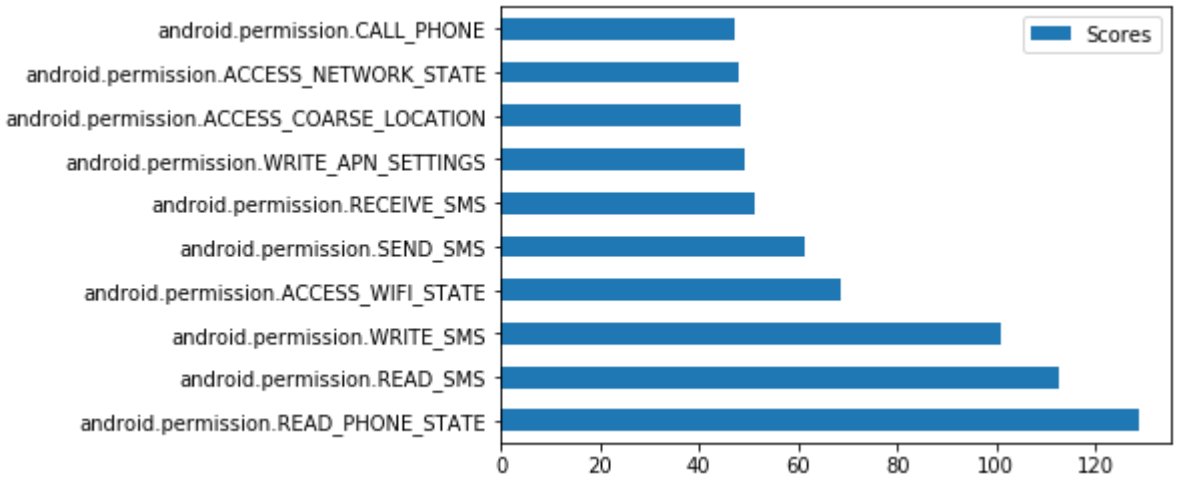
```
 Evaluation for estimator ***********  random_forest
Pipeline(memory=None,
     steps=[('clf', RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=30,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=-1,
            oob_score=True, random_state=50, verbose=0, warm_start=False))])
Accuracy is =  0.9125

 Evaluation for estimator ***********  xgboost
Pipeline(memory=None,
     steps=[('clf', XGBClassifier(base_score=None, booster=None, colsample_byl
evel=None,
       colsample_bynode=None, colsample_bytree=None, gamma=None,
       gpu_id=None, importance_type='gain', interaction_constraints=None,
       learning_rate=None, max_delta_step=None, max_depth=None,
       min_ch...
       tree_method=None, use_label_encoder=True, validate_parameters=None,
       verbosity=None))])
[17:44:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.
3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation met
ric used with the objective 'binary:logistic' was changed from 'error' to 'log
loss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy is =  0.925

 Evaluation for estimator ***********  svc
Pipeline(memory=None,
     steps=[('clf', SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False))])
Accuracy is =  0.8875
Chosen best estimator =  xgboost best accuracy is =  0.925
[17:44:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.
3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation met
ric used with the objective 'binary:logistic' was changed from 'error' to 'log
loss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

ML1 + FE1

In [50]:
```
tr = Trainer(estimator="random_forest", model_name="malware_detection_no_fe", choo
tr.train()
tr.eval()
```

```
Number of columns with greater than 0.4 correlation are 57
Columns with greater than 0.4 correlation are {'android.permission.DEVICE_POWE
R', 'android.permission.RECORD_AUDIO', 'android.permission.BLUETOOTH_ADMIN',
'android.permission.FLASHLIGHT', 'android.permission.WRITE_SYNC_SETTINGS', 'an
droid.permission.SEND_SMS', 'com.android.launcher.permission.INSTALL_SHORTCU
T', 'android.permission.READ_LOGS', 'android.permission.MODIFY_PHONE_STATE',
'android.permission.READ_SYNC_SETTINGS', 'android.permission.RECEIVE_MMS', 'co
m.android.launcher.permission.UNINSTALL_SHORTCUT', 'android.permission.RESTART
_PACKAGES', 'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.US
E_CREDENTIALS', 'android.permission.READ_CALENDAR', 'android.permission.ACCESS
_DOWNLOAD_MANAGER', 'android.permission.READ_SYNC_STATS', 'android.permission.
ACCESS_DOWNLOAD_MANAGER_ADVANCED', 'android.permission.INTERNET', 'android.per
mission.GET_PACKAGE_SIZE', 'android.permission.WRITE_APN_SETTINGS', 'android.p
ermission.RECEIVE_WAP_PUSH', 'com.android.launcher.permission.WRITE_SETTINGS',
'android.permission.WRITE_CALENDAR', 'android.permission.GET_ACCOUNTS', 'andro
id.permission.DISABLE_KEYGUARD', 'android.permission.READ_SMS', 'android.permi
ssion.ACCESS_FINE_LOCATION', 'android.permission.BLUETOOTH', 'android.permissi
on.ACCESS_LOCATION_EXTRA_COMMANDS', 'android.permission.SET_WALLPAPER_HINTS',
'android.permission.WRITE_CONTACTS', 'android.permission.PROCESS_OUTGOING_CALL
S', 'android.permission.SET_WALLPAPER', 'android.permission.INJECT_EVENTS', 'a
ndroid.permission.GLOBAL_SEARCH_CONTROL', 'android.permission.RECEIVE_SMS', 'a
ndroid.permission.PERSISTENT_ACTIVITY', 'android.permission.SEND_DOWNLOAD_COMP
LETED_INTENTS', 'android.permission.READ_CONTACTS', 'android.permission.WRITE_
SMS', 'android.permission.HARDWARE_TEST', 'android.permission.UPDATE_DEVICE_ST
ATS', 'com.android.browser.permission.WRITE_HISTORY_BOOKMARKS', 'android.permi
ssion.WRITE_SECURE_SETTINGS', 'android.permission.READ_PHONE_STATE', 'android.
permission.INSTALL_PACKAGES', 'android.permission.CHANGE_WIMAX_STATE', 'androi
d.permission.READ_CALL_LOG', 'android.permission.WRITE_SETTINGS', 'android.per
mission.ACCESS_WIFI_STATE', 'android.permission.DELETE_CACHE_FILES', 'com.andr
oid.launcher.permission.READ_SETTINGS', 'android.permission.REBOOT', 'android.
permission.VIBRATE', 'android.permission.MANAGE_ACCOUNTS'}

 Top 10 important features are
                                          Specs      Scores
190        android.permission.READ_PHONE_STATE  128.766355
195                android.permission.READ_SMS  112.500000
277               android.permission.WRITE_SMS  101.038095
26        android.permission.ACCESS_WIFI_STATE   68.512195
225                android.permission.SEND_SMS   61.493151
208             android.permission.RECEIVE_SMS   51.428571
266      android.permission.WRITE_APN_SETTINGS   49.075472
9    android.permission.ACCESS_COARSE_LOCATION   48.268817
22     android.permission.ACCESS_NETWORK_STATE   48.144105
76               android.permission.CALL_PHONE   47.032258

 Accuracy is 0.925
Precision 0.925
Recall 0.925

 Test Classification Report
              precision    recall  f1-score   support

           0       0.88      0.97      0.93        38
```

```
          1       0.97      0.88      0.93        42

avg / total       0.93      0.93      0.93        80


 Train Classification Report
              precision    recall  f1-score   support

          0       0.99      1.00      0.99       161
          1       1.00      0.99      0.99       157

avg / total       0.99      0.99      0.99       318


 Test Confusion Matrix
 [[37  1]
 [ 5 37]]

 Train Confusion Matrix
 [[161   0]
 [  2 155]]
```

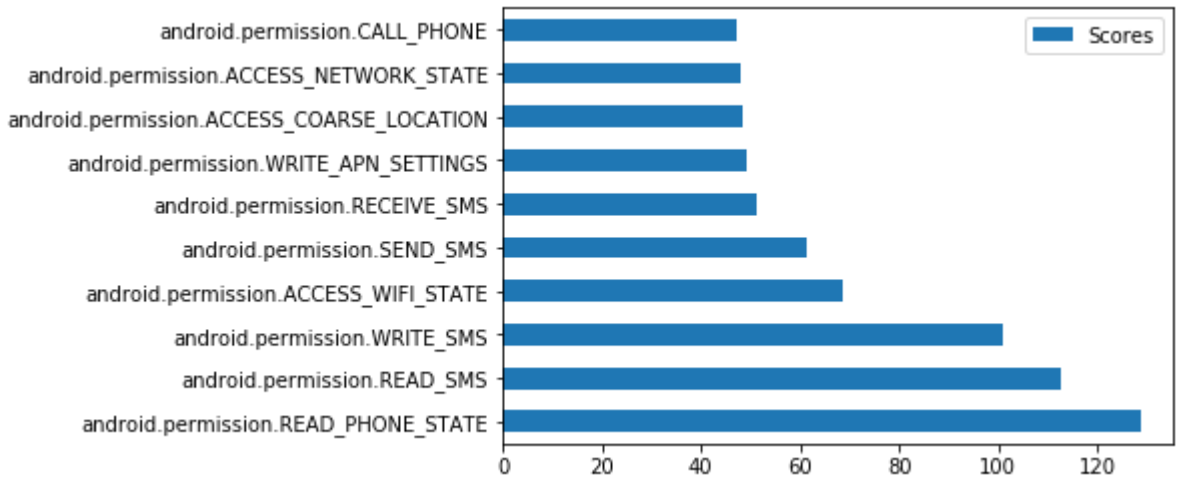


ML2 + FE1

In [51]:
```
tr = Trainer(estimator="xgboost", model_name="malware_detection_no_fe", choose_bes
tr.train()
tr.eval()
```

```
Number of columns with greater than 0.4 correlation are 57
Columns with greater than 0.4 correlation are {'android.permission.DEVICE_POWE
R', 'android.permission.RECORD_AUDIO', 'android.permission.BLUETOOTH_ADMIN',
'android.permission.FLASHLIGHT', 'android.permission.WRITE_SYNC_SETTINGS', 'an
droid.permission.SEND_SMS', 'com.android.launcher.permission.INSTALL_SHORTCU
T', 'android.permission.READ_LOGS', 'android.permission.MODIFY_PHONE_STATE',
'android.permission.READ_SYNC_SETTINGS', 'android.permission.RECEIVE_MMS', 'co
m.android.launcher.permission.UNINSTALL_SHORTCUT', 'android.permission.RESTART
_PACKAGES', 'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.US
E_CREDENTIALS', 'android.permission.READ_CALENDAR', 'android.permission.ACCESS
_DOWNLOAD_MANAGER', 'android.permission.READ_SYNC_STATS', 'android.permission.
ACCESS_DOWNLOAD_MANAGER_ADVANCED', 'android.permission.INTERNET', 'android.per
mission.GET_PACKAGE_SIZE', 'android.permission.WRITE_APN_SETTINGS', 'android.p
ermission.RECEIVE_WAP_PUSH', 'com.android.launcher.permission.WRITE_SETTINGS',
'android.permission.WRITE_CALENDAR', 'android.permission.GET_ACCOUNTS', 'andro
id.permission.DISABLE_KEYGUARD', 'android.permission.READ_SMS', 'android.permi
ssion.ACCESS_FINE_LOCATION', 'android.permission.BLUETOOTH', 'android.permissi
on.ACCESS_LOCATION_EXTRA_COMMANDS', 'android.permission.SET_WALLPAPER_HINTS',
'android.permission.WRITE_CONTACTS', 'android.permission.PROCESS_OUTGOING_CALL
S', 'android.permission.SET_WALLPAPER', 'android.permission.INJECT_EVENTS', 'a
ndroid.permission.GLOBAL_SEARCH_CONTROL', 'android.permission.RECEIVE_SMS', 'a
ndroid.permission.PERSISTENT_ACTIVITY', 'android.permission.SEND_DOWNLOAD_COMP
LETED_INTENTS', 'android.permission.READ_CONTACTS', 'android.permission.WRITE_
SMS', 'android.permission.HARDWARE_TEST', 'android.permission.UPDATE_DEVICE_ST
ATS', 'com.android.browser.permission.WRITE_HISTORY_BOOKMARKS', 'android.permi
ssion.WRITE_SECURE_SETTINGS', 'android.permission.READ_PHONE_STATE', 'android.
permission.INSTALL_PACKAGES', 'android.permission.CHANGE_WIMAX_STATE', 'androi
d.permission.READ_CALL_LOG', 'android.permission.WRITE_SETTINGS', 'android.per
mission.ACCESS_WIFI_STATE', 'android.permission.DELETE_CACHE_FILES', 'com.andr
oid.launcher.permission.READ_SETTINGS', 'android.permission.REBOOT', 'android.
permission.VIBRATE', 'android.permission.MANAGE_ACCOUNTS'}

 Top 10 important features are
                                          Specs      Scores
190        android.permission.READ_PHONE_STATE  128.766355
195                android.permission.READ_SMS  112.500000
277               android.permission.WRITE_SMS  101.038095
26        android.permission.ACCESS_WIFI_STATE   68.512195
225                android.permission.SEND_SMS   61.493151
208             android.permission.RECEIVE_SMS   51.428571
266      android.permission.WRITE_APN_SETTINGS   49.075472
9    android.permission.ACCESS_COARSE_LOCATION   48.268817
22     android.permission.ACCESS_NETWORK_STATE   48.144105
76               android.permission.CALL_PHONE   47.032258
[11:54:36] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.
3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation met
ric used with the objective 'binary:logistic' was changed from 'error' to 'log
loss'. Explicitly set eval_metric if you'd like to restore the old behavior.

 Accuracy is 0.8875
Precision 0.8875
Recall 0.8875
```

```
 Test Classification Report
             precision    recall  f1-score   support

          0       0.87      0.89      0.88        38
          1       0.90      0.88      0.89        42

avg / total       0.89      0.89      0.89        80


 Train Classification Report
             precision    recall  f1-score   support

          0       0.99      1.00      1.00       161
          1       1.00      0.99      1.00       157

avg / total       1.00      1.00      1.00       318


 Test Confusion Matrix
 [[34  4]
 [ 5 37]]

 Train Confusion Matrix
 [[161   0]
 [  1 156]]
```



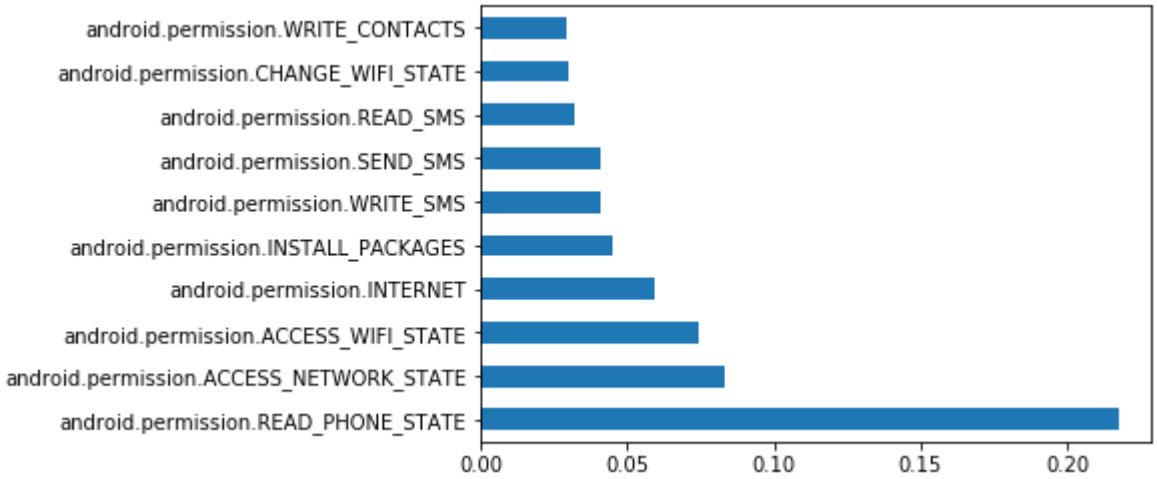
ML1 + FE2

In [52]:
```
tr = Trainer(estimator="random_forest", model_name="malware_detection_no_fe", choo
tr.train()
tr.eval()
```

```
Number of columns with greater than 0.6 correlation are 41
Columns with greater than 0.6 correlation are {'android.permission.BLUETOOTH_A
DMIN', 'android.permission.FLASHLIGHT', 'android.permission.CHANGE_WIFI_STAT
E', 'android.permission.WRITE_SYNC_SETTINGS', 'android.permission.SEND_SMS',
'android.permission.MODIFY_PHONE_STATE', 'android.permission.READ_SYNC_SETTING
S', 'android.permission.RECEIVE_MMS', 'android.permission.RESTART_PACKAGES',
'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.USE_CREDENTIAL
S', 'android.permission.ACCESS_DOWNLOAD_MANAGER', 'android.permission.READ_SYN
C_STATS', 'android.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED', 'android.perm
ission.INTERNET', 'android.permission.WRITE_APN_SETTINGS', 'android.permissio
n.RECEIVE_WAP_PUSH', 'com.android.launcher.permission.WRITE_SETTINGS', 'androi
d.permission.WRITE_CALENDAR', 'android.permission.DISABLE_KEYGUARD', 'android.
permission.READ_SMS', 'android.permission.ACCESS_FINE_LOCATION', 'android.perm
ission.SET_WALLPAPER_HINTS', 'android.permission.WRITE_CONTACTS', 'android.per
mission.PROCESS_OUTGOING_CALLS', 'android.permission.INJECT_EVENTS', 'android.
permission.GLOBAL_SEARCH_CONTROL', 'android.permission.RECEIVE_SMS', 'android.
permission.PERSISTENT_ACTIVITY', 'android.permission.SEND_DOWNLOAD_COMPLETED_I
NTENTS', 'android.permission.READ_CONTACTS', 'android.permission.HARDWARE_TES
T', 'android.permission.UPDATE_DEVICE_STATS', 'com.android.browser.permission.
WRITE_HISTORY_BOOKMARKS', 'android.permission.WRITE_SECURE_SETTINGS', 'androi
d.permission.CHANGE_WIMAX_STATE', 'android.permission.WRITE_SETTINGS', 'androi
d.permission.DELETE_CACHE_FILES', 'com.android.launcher.permission.READ_SETTIN
GS', 'android.permission.REBOOT', 'android.permission.MANAGE_ACCOUNTS'}

 Top 10 important features are
 android.permission.READ_PHONE_STATE        0.217818
android.permission.ACCESS_NETWORK_STATE    0.082906
android.permission.ACCESS_WIFI_STATE       0.074627
android.permission.INTERNET                0.059054
android.permission.INSTALL_PACKAGES        0.045282
android.permission.WRITE_SMS               0.041158
android.permission.SEND_SMS                0.040992
android.permission.READ_SMS                0.031738
android.permission.CHANGE_WIFI_STATE       0.030113
android.permission.WRITE_CONTACTS          0.029048
dtype: float64

 Accuracy is 0.925
Precision 0.925
Recall 0.925

 Test Classification Report
             precision    recall  f1-score   support

          0       0.88      0.97      0.93        38
          1       0.97      0.88      0.93        42

avg / total       0.93      0.93      0.93        80


 Train Classification Report
             precision    recall  f1-score   support
```

0	0.98	1.00	0.99	161
1	1.00	0.98	0.99	157
avg / total	0.99	0.99	0.99	318

```
Test Confusion Matrix
[[37  1]
[ 5 37]]

Train Confusion Matrix
[[161   0]
[  3 154]]
```

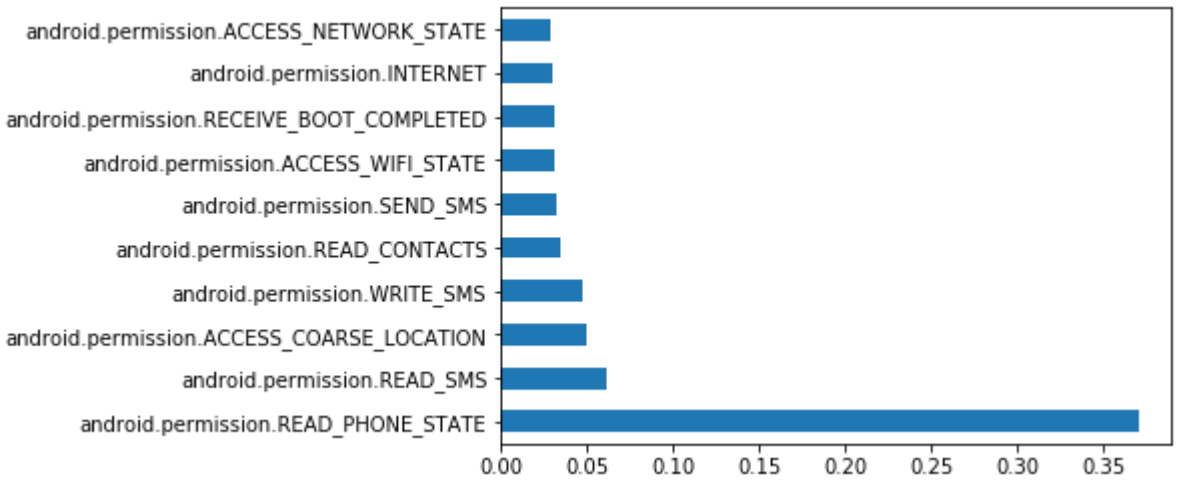


ML2 + FE2

In [53]:
```
tr = Trainer(estimator="xgboost", model_name="malware_detection_no_fe", choose_bes
tr.train()
tr.eval()
```

```
Number of columns with greater than 0.6 correlation are 41
Columns with greater than 0.6 correlation are {'android.permission.BLUETOOTH_A
DMIN', 'android.permission.FLASHLIGHT', 'android.permission.CHANGE_WIFI_STAT
E', 'android.permission.WRITE_SYNC_SETTINGS', 'android.permission.SEND_SMS',
'android.permission.MODIFY_PHONE_STATE', 'android.permission.READ_SYNC_SETTING
S', 'android.permission.RECEIVE_MMS', 'android.permission.RESTART_PACKAGES',
'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.USE_CREDENTIAL
S', 'android.permission.ACCESS_DOWNLOAD_MANAGER', 'android.permission.READ_SYN
C_STATS', 'android.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED', 'android.perm
ission.INTERNET', 'android.permission.WRITE_APN_SETTINGS', 'android.permissio
n.RECEIVE_WAP_PUSH', 'com.android.launcher.permission.WRITE_SETTINGS', 'androi
d.permission.WRITE_CALENDAR', 'android.permission.DISABLE_KEYGUARD', 'android.
permission.READ_SMS', 'android.permission.ACCESS_FINE_LOCATION', 'android.perm
ission.SET_WALLPAPER_HINTS', 'android.permission.WRITE_CONTACTS', 'android.per
mission.PROCESS_OUTGOING_CALLS', 'android.permission.INJECT_EVENTS', 'android.
permission.GLOBAL_SEARCH_CONTROL', 'android.permission.RECEIVE_SMS', 'android.
permission.PERSISTENT_ACTIVITY', 'android.permission.SEND_DOWNLOAD_COMPLETED_I
NTENTS', 'android.permission.READ_CONTACTS', 'android.permission.HARDWARE_TES
T', 'android.permission.UPDATE_DEVICE_STATS', 'com.android.browser.permission.
WRITE_HISTORY_BOOKMARKS', 'android.permission.WRITE_SECURE_SETTINGS', 'androi
d.permission.CHANGE_WIMAX_STATE', 'android.permission.WRITE_SETTINGS', 'androi
d.permission.DELETE_CACHE_FILES', 'com.android.launcher.permission.READ_SETTIN
GS', 'android.permission.REBOOT', 'android.permission.MANAGE_ACCOUNTS'}

 Top 10 important features are
 android.permission.READ_PHONE_STATE          0.371109
android.permission.READ_SMS                  0.060936
android.permission.ACCESS_COARSE_LOCATION    0.049157
android.permission.WRITE_SMS                 0.047497
android.permission.READ_CONTACTS             0.034802
android.permission.SEND_SMS                  0.031870
android.permission.ACCESS_WIFI_STATE         0.030956
android.permission.RECEIVE_BOOT_COMPLETED    0.030892
android.permission.INTERNET                  0.030405
android.permission.ACCESS_NETWORK_STATE      0.028556
dtype: float64
[11:54:53] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.
3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation met
ric used with the objective 'binary:logistic' was changed from 'error' to 'log
loss'. Explicitly set eval_metric if you'd like to restore the old behavior.

 Accuracy is 0.9
Precision 0.9
Recall 0.9

 Test Classification Report
              precision    recall  f1-score   support

           0       0.88      0.92      0.90        38
           1       0.93      0.88      0.90        42

avg / total       0.90      0.90      0.90        80
```

```
Train Classification Report
             precision    recall  f1-score   support

          0       0.99      0.99      0.99       161
          1       0.99      0.99      0.99       157

avg / total       0.99      0.99      0.99       318


Test Confusion Matrix
[[35  3]
[ 5 37]]

Train Confusion Matrix
[[160   1]
[  2 155]]
```



Comparison

Based on the different runs implemented above, we can see that Random forest classifier works best with the data when both feature engineering techniques are applied.

Machine Learning Technique	Feature Engineering Technique	Accuracy
Random Forest (ML1)	Correlation (0.4) + SelectKBest (FE1)	0.925
XGBoost (ML2)	Correlation (0.4) + SelectKBest (FE1)	0.8875
Random Forest (ML1)	Correlation (0.6) + ExtraTreesClassifier (FE2)	0.925
XGBoost (ML2)	Correlation (0.6) + ExtraTreesClassifier (FE2)	0.9

In [ ]: