Background

Machine Learning, integrated with Information Systems helps people make decisions in various areas like medical diagnosis, loan-granting, and areas which may have ethical and legal implications. Therefore, while building such systems, it is necessary to think beyond the objective of maximizing prediction accuracy and verify if the predictions of such systems are 'fair' (unbiased).

If ML models are biased, then such unfairness is introduced either due to prejudice or negative legacy, which is depicted when data has some sensitive attributes like gender, race, age category. Such sensitve attributes divide the individuals into groups. So, it required to build a model which is not only **Fairness Aware** but also mitigate the unfairness. In this project we are trying to demonstrate the unfairness/ bias nature of models (due to direct/ indirect biased historical data) and will apply some fairness aware methods to model for mitigating the bias factor.

*Note: During the course of this project we will use the word **Protected Attributes** to refer sensitive attributes.*

Outline

There is an increasing reliance on algorithms to assess a defendant's likelihood of becoming a recidivist (a term used to describe convicted criminals who reoffend). These risk assessment scores significantly impact the severity of the sentence for defendants. For example, Broward County in Florida uses the COMPAS scoring algorithm to help determine pretrial release decisions.

Propublica perform an analysis of the COMPAS recidivism algorithm for defendants in Broward County and revealed that COMPAS scores are biased against black defendants. ProPublica published their methodology for measuring this bias.**[1]**

In this project, we perform our own exploration and modeling of criminal risk scores. To do that we built our own model that uses the existing features from an individual's profile to predict the likelihood of recidivating (a model that is not created in ProPublica's analysis). We implemented our own Logistic Regression, Decision Tree and Random Forest model from COMPAS dataset and applied various fairness-aware methods to reduce the bias factor. We used Themis-ml(an open source machine learning library that implements several fairness-aware methods)for our study **[2]**. We concluded our study with comparing different fair-aware methods and discussing about Fairness Utility Trade-Off.

Fairness Aware methods used:

1) Baseline (B): Train the model on all available input variables, including protected attributes

2) Remove Protected Attribute (RPA): Train a model on input variables without protected attributes. This is naive fairness-aware approach

3) Relabel Target Variable (RTV): Train a model using the Relabelling fairness-aware method

4) Additive Counterfactually Fair Model (ACF): Train a model using the Additive Counterfactually Fair method

5) Reject Option Classification (ROC): Train a model using Reject Option Classifcation method

In [1]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
import math
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import matplotlib.patches as patches
from collections import Counter
from sklearn.metrics import confusion_matrix
import itertools
from sklearn.metrics import precision_score
from sklearn.metrics import roc_curve
import warnings
warnings.filterwarnings("ignore")
```

```
/anaconda3/lib/python3.6/site-packages/statsmodels/compat/pandas.p
y:56: FutureWarning: The pandas.core.datetools module is deprecate
d and will be removed in a future version. Please use the pandas.t
series module instead.
  from pandas.core import datetools
```

In [2]:

```
import pandas as pd
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

General Recidivism Data of two years

In [3]:

```
raw_data = pd.read_csv("compas-scores-two-years.csv")
```

Data Description

For this study we use the dataset made available by ProPublica. Manual exploration of compas.db help us to come up with a data dictonary containing best guess of what each attribute depicts.

Data dictionary containing description of the variables we used in our model

Variable	**Description**
compas_screening_date	date on which decile_score was given
sex	sex (male or female)
dob	date of birth
age	age in years
age_cat	age category (less than 25, 25-45, greater than 45)
race	race (African-American, Asian, Caucasian, Hispanic, Native American, Other)
juv_fel_count	juvenile felony count
decile_score	COMPAS Risk of Recidivism score from 1 to 10
juv_misd_count	juvenile misdemeanor count
juv_other_count	juvenile other offenses count
priors_count	prior offenses count
days_b_screening_arrest	number of days between COMPAS screening and arrest
c_jail_in	jail entry date for original crime
c_jail_out	jail exit date for original crime
c_arrest_date	arrest date for original crime
c_days_from_compas	days between COMPAS screening and original crime offense date
c_charge_degree	charge degree of original crime
is_recid	binary indicator of recidivation (1=individual recidivated, 0=individual did not recidivate)
score_text	ProPublica-defined category of decile_score (High=8-10, Medium=5-7, Low=1-4)

Data Preparation

Dropping the columns which we are not using for this analysis, cleaning data by removing null values.

- In order to match COMPAS scores with the corresponding recent cases, we considered cases with arrest dates or charge dates within 30 days of a COMPAS assessment being conducted.
- We are not considering ordinary crimes like tickets, failure to show in court etc.

In [5]:

```
df = raw_data.drop(['c_offense_date','c_case_number','c_arrest_date','r_case_n
umber','r_charge_degree',
                    'r_days_from_arrest','r_offense_date','r_charge_desc','r_j
ail_in','r_jail_out','violent_recid',
                    'decile_score.1','priors_count.1','start','end','vr_case_n
umber','vr_charge_degree',
                    'vr_offense_date','vr_charge_desc','two_year_recid'],axis=
1)
```

In [6]:

```
df.dropna(inplace=True)
```

In [7]:

```
df = df[(df['days_b_screening_arrest'] <= 30) & (df.days_b_screening_arrest >=
-30) &
                (df.c_charge_degree != "O") & (df.score_text != 'N/A')]
```

In [8]:

```
df.shape
```

Out[8]:

```
(6167, 33)
```

This method is used to translate score text into two binary values (0,1) where High, Medium is assigned as 1 and Low is assigned as 0.

In [4]:

```
def low_high(df):
    df.ix[(df.score_text == 'Medium'), 'score_factor'] = 1
    df.ix[(df.score_text == 'Low'), 'score_factor'] = 0
    df.ix[(df.score_text == 'High'), 'score_factor'] = 1
    return df
```

We need to include number of 'hours' spent in jail so below calculation takes in jail_in and jail_out for the original crime and determine the number of hours spent in jail

In [9]:

```
df['c_jail_in'] = pd.to_datetime(df['c_jail_in'])
```

In [10]:

```
df['c_jail_out'] = pd.to_datetime(df['c_jail_out'])
```

In [11]:

```
df['c_num_hours'] = (df['c_jail_out'] - df['c_jail_in']).astype('timedelta64[h
]')
```

Calculate number of hours spent in custody.

In [12]:

```
df['in_custody'] = pd.to_datetime(df['in_custody'])
```

In [13]:

```
df['out_custody'] = pd.to_datetime(df['out_custody'])
```

In [14]:

```
df['num_hours_custody'] = (df['out_custody'] - df['in_custody']).astype('timed
elta64[h]')
```

Creating dummies for three variables - gender, race and age_category.

> We have classified race, gender and age_category as protected classes/ attributes. Hence, among them we need to classify advantage and disadvantage group[as per themis-ml]. Thus, we have added new columns,
>
> 1) is_female which is a binary column for gender containing values 0 for not female and 1 for female. Hence, advantage group is not female and disadvantage group is Female.
> 2) is_African_American : This column is for race but divided into a binary category where either a person is African-American or not. Reason to choose African American is that the data is more for African_American and Caucasians and very less data for Hispanic and others. So, our norm is, advantage group = 0 [not African American] , disadvantage group = 1[African American].
> 3) is_less_than_25 age category : This has three different categories and to make it binary we have divided it into a binary category(less than 25). So, our norm is: less than 25 = 1 (disadvantage group) and greater than 25 = 0(advantage group)

In [15]:

```
df =  pd.concat([df,pd.get_dummies(df['sex'],prefix="is")], axis=1)
```

In [16]:

```
df = pd.concat([df,pd.get_dummies(df['race'],prefix="is")], axis=1)
```

In [17]:

```
df = pd.concat([df,pd.get_dummies(df['age_cat'],prefix="is")], axis=1)
```

In [18]:

```
df.head(2)
```

Out[18]:

	id	name	first	last	compas_screening_date	sex	dob	age	age_
0	1	miguel hernandez	miguel	hernandez	2013-08-14	Male	1947-04-18	69	Grea than
1	3	kevon dixon	kevon	dixon	2013-01-27	Male	1982-01-22	34	25 -

Dropping the columns which are not required for analysis

In [19]:

```
df_under_test = df.drop(['id','name','first','last','compas_screening_date','d
ob','c_jail_in','c_jail_out',
                         'c_charge_desc','is_violent_recid','type_of_assessment'
,'screening_date',
                         'v_type_of_assessment','v_decile_score','v_score_text',
'v_screening_date','in_custody',
                         'out_custody','event','is_Male','is_Asian',
                          'is_Caucasian','is_Hispanic','is_Native American','is_
Other','is_25 - 45',
                         'is_Greater than 45'],axis=1)
```

In [20]:

```
df_under_test.head(2)
```

Out[20]:

	sex	age	age_cat	race	juv_fel_count	decile_score	juv_misd_count	juv_o
0	Male	69	Greater than 45	Other	0	1	0	0
1	Male	34	25 - 45	African-American	0	3	0	0

Crime is categorised into 1 and 0

1 for Felony and 0 for misdemeanor

In [21]:

```
def convert_charge_degree_to_binary(df):
    df.ix[(df.c_charge_degree == 'F'), 'crime_factor'] = 1
    df.ix[(df.c_charge_degree == 'M'), 'crime_factor'] = 0
    return df
```

In [22]:

```
dut_final = convert_charge_degree_to_binary(df_under_test)
```

In [23]:

```
dut_final = low_high(dut_final)
```

In [24]:

```
dut_final.head(2)
```

Out[24]:

	sex	age	age_cat	race	juv_fel_count	decile_score	juv_misd_count	juv_o
0	Male	69	Greater than 45	Other	0	1	0	0
1	Male	34	25 - 45	African-American	0	3	0	0

In [25]:

```
new_data = dut_final[['juv_fel_count','decile_score','juv_misd_count','juv_oth
er_count','priors_count',
                       'days_b_screening_arrest','c_days_from_compas','age'
                       ,'c_num_hours','num_hours_custody','is_recid',
                       'crime_factor','score_factor','is_Female','is_African-A
merican','is_Less than 25']]
```

In [26]:

```
new_data.head(2)
```

Out[26]:

	juv_fel_count	decile_score	juv_misd_count	juv_other_count	priors_count	days
0	0	1	0	0	0	-1.0
1	0	3	0	0	0	-1.0

Scaling the data columns

In [27]:

```
scaled_features = new_data.copy()
col_names = ['age', 'c_num_hours', 'num_hours_custody', 'days_b_screening_arre
st']
features = scaled_features[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
scaled_features[col_names] = features
```

In [28]:

```
scaled_features.head(3)
```

Out[28]:

	juv_fel_count	decile_score	juv_misd_count	juv_other_count	priors_count	days
0	0	1	0	0	0	0.14
1	0	3	0	0	0	0.14
2	0	4	0	1	4	0.14

In [29]:

```
scaled_features.columns
```

Out[29]:

```
Index(['juv_fel_count', 'decile_score', 'juv_misd_count', 'juv_oth
er_count', 'priors_count', 'days_b_screening_arrest', 'c_days_from
_compas', 'age', 'c_num_hours', 'num_hours_custody', 'is_recid', '
crime_factor', 'score_factor', 'is_Female', 'is_African-American',
'is_Less than 25'], dtype='object')
```

In [30]:

```
scaled_features.shape
```

Out[30]:

```
(6167, 16)
```

Since, desired outcome for a defendant is : "not to recidivate". So, created a new column 'will_not_recidivate' which is inverse of is_recid. In 'will_not_recidivate' we define 0 = defendant will recidivate & 1 = defendant will not recidivate.

In [31]:

```
def will_not_recidivate(df):
    df.ix[(df.is_recid == 1), 'will_not_recidivate'] = 0
    df.ix[(df.is_recid == 0), 'will_not_recidivate'] = 1
    return df
```

In [32]:

```
scaled_features = will_not_recidivate(scaled_features)
```

In [33]:

```
columns_under_test = ['juv_fel_count', 'decile_score', 'juv_misd_count', 'juv_
other_count', 'priors_count',
                      'days_b_screening_arrest', 'c_days_from_compas', 'age',
'c_num_hours', 'num_hours_custody',
                      'crime_factor', 'score_factor', 'is_Female', 'is_African
-American', 'is_Less than 25']
```

Methods to apply model

This section contains all the fairness aware methods including Baseline and related methods used to calculate the two important measures, `mean difference` and `accuracy`.

Mean Difference [2] [3], measures the degree to which data or predictions is potentially discriminated. Mean difference measures the difference between the means of the targets of the protected group and the general group, $d = E(y^+|s_0) - E(y^+|s_1)$. If there is not difference then it is considered that there is no discrimination. The values of mean difference varies from between (-1, +1), were -1 is reverse discrimination case and +1 is full discrimintation case.

Accuracy, measures the predictive power of a model.

In [34]:

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

from sklearn.model_selection import StratifiedKFold, RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from themis_ml.metrics import mean_difference
from sklearn.metrics import (
    accuracy_score, roc_auc_score, f1_score)
```

In [35]:

```
N_SPLITS = 10
N_REPEATS = 5
RANDOM_STATE = 1000

def get_estimator_name(e):
    return "".join([x for x in str(type(e)).split(".")[-1]
                    if x.isalpha()])



def get_grid_params(grid_params_dict):
    """Get outer product of grid search parameters."""
    return [
        dict(params) for params in itertools.product(
            *[[(k, v_i) for v_i in v] for
              k, v in grid_params_dict.items()])]

def fit_with_s(estimator):
    has_relabeller = getattr(estimator, "relabeller", None) is not None
    child_estimator = getattr(estimator, "estimator", None)
    estimator_fit_with_s = getattr(estimator, "S_ON_FIT", False)
    child_estimator_fit_with_s = getattr(child_estimator, "S_ON_FIT", False)
    return has_relabeller or estimator_fit_with_s or\
        child_estimator_fit_with_s



def predict_with_s(estimator):
    estimator_pred_with_s = getattr(estimator, "S_ON_PREDICT", False)
    child_estimator = getattr(estimator, "estimator", None)
    return estimator_pred_with_s or \
        getattr(child_estimator, "S_ON_PREDICT", False)



def cross_validation_experiment(estimators, X, y, s, s_name):
    msg = "Training models: protected_class = %s" % s_name
    print(msg)
    print("-" * len(msg))
    performance_scores = []
    # stratified groups tries to balance out y and s
    groups = [i + j for i, j in
              zip(y.astype(str), s_female.astype(str))]
    cv = RepeatedStratifiedKFold(

        n_splits=N_SPLITS,
        n_repeats=N_REPEATS,
        random_state=RANDOM_STATE)
    for e_name, e in estimators:
        print("%s, fold:" % e_name)
        for i, (train, test) in enumerate(cv.split(X, y, groups=groups)):
            # create train and validation fold partitions
            X_train, X_test = X[train], X[test]
            y_train, y_test = y[train], y[test]
            s_train, s_test = s[train], s[test]

            # fit model and generate train and test predictions
            if fit_with_s(e):
                e.fit(X_train, y_train, s_train)
            else:
                e.fit(X_train, y_train)

            train_pred_args = (X_train, s_train) if predict_with_s(e) \
                else (X_train, )
            test_pred_args = (X_test, s_test) if predict_with_s(e) \
                else (X_test, )

            train_pred_prob = e.predict_proba(*train_pred_args)[:, 1]
            train_pred = e.predict(*train_pred_args)
            test_pred_prob = e.predict_proba(*test_pred_args)[:, 1]
            test_pred = e.predict(*test_pred_args)
            train_accuracy = accuracy_score(y_train, train_pred)
            test_accuracy = accuracy_score(y_test, test_pred)
            # train scores
            performance_scores.append([
                s_name, e_name, i, "train",
                # fairness metrics
                mean_difference(train_pred, s_train)[0],
                # accuracy
                train_accuracy,
            ])
            # test scores
            performance_scores.append([
                s_name, e_name, i, "test",
                # fairness metrics
                mean_difference(test_pred, s_test)[0],
                # accuracy
                test_accuracy,
            ])
    return pd.DataFrame(
        performance_scores,
        columns=[
            "protected_class", "estimator", "cv_fold", "fold_type",
            "mean_diff", "accuracy"])
```

Baseline (B)

Train our models on all available input variables, including protected attributes. This will serve as our baseline comparison with other fairness aware methods

In [36]:

```
b_X = scaled_features[columns_under_test].values
```

In [37]:

```
y = scaled_features['will_not_recidivate'].values
```

In [38]:

```
s_female = scaled_features['is_Female'].values
s_black = scaled_features["is_African-American"].values
s_age_less_than_25 = scaled_features["is_Less than 25"].values
```

In [39]:

```
LOGISTIC_REGRESSION = LogisticRegression(
    penalty="l2", C=0.001, class_weight="balanced")
DECISION_TREE_CLF = DecisionTreeClassifier(
    criterion="entropy", max_depth=10, min_samples_leaf=10, max_features=15,
    class_weight="balanced")
RANDOM_FOREST_CLF = RandomForestClassifier(
    criterion="entropy", n_estimators=50, max_depth=10, max_features=15,
    min_samples_leaf=10, class_weight="balanced")
estimators = [
    ("LogisticRegression", LOGISTIC_REGRESSION),
    ("DecisionTree", DECISION_TREE_CLF),
    ("RandomForest", RANDOM_FOREST_CLF)
]
```

In [40]:

```
experiment_baseline_female = cross_validation_experiment(
    estimators, b_X, y, s_female, "GENDER (Male=0, Female=1)")
experiment_baseline_race = cross_validation_experiment(
    estimators, b_X, y, s_black, "RACE (Other=0, AA=1)")
experiment_baseline_age = cross_validation_experiment(
    estimators, b_X, y, s_age_less_than_25, "AGE CATEGORY (Other=0, Below 25=1
)")
```

```
Training models: protected_class = GENDER (Male=0, Female=1)
------------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = RACE (Other=0, AA=1)
-------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = AGE CATEGORY (Other=0, Below 25
=1)
------------------------------------------------------------------
---
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
```

In [41]:

```
import seaborn as sns
import matplotlib.pyplot as plt
% matplotlib inline

FAIRNESS_METRICS = ["mean_diff"]
ACCURACY_METRICS = ["accuracy"]

def summarize_experiment_results(experiment_df):
    return (
        experiment_df
        .drop("cv_fold", axis=1)
        .groupby(["protected_class", "estimator", "fold_type"])
        .mean())

experiment_baseline = pd.concat([
    experiment_baseline_female,
    experiment_baseline_race,
    experiment_baseline_age
])
experiment_baseline_summary = summarize_experiment_results(
    experiment_baseline)
experiment_baseline_summary.query("fold_type == 'test'")
```

Out[41]:

			mean_diff	accuracy
protected_class	**estimator**	**fold_type**		
AGE CATEGORY (Other=0, Below 25=1)	**DecisionTree**	**test**	0.159953	0.684645
	LogisticRegression	**test**	0.406302	0.679812
	RandomForest	**test**	0.215616	0.714805
GENDER (Male=0, Female=1)	**DecisionTree**	**test**	-0.149723	0.684937
	LogisticRegression	**test**	-0.190379	0.679812
	RandomForest	**test**	-0.169214	0.714706
RACE (Other=0, AA=1)	**DecisionTree**	**test**	0.203724	0.685261
	LogisticRegression	**test**	0.229384	0.679812
	RandomForest	**test**	0.239449	0.713637

Summary Baseline:

When all the protected attributes were present we observe the following results:

1. For Protected Atrribute, Age Category, `Decision Tree` classifier has the least mean difference of 15.96% with an accuracy of 68.5%. So, as per model, people below age 25 are almost 16% more likely to be recidivated than other age group people.

2. For Protected Attribute, Gender, `Decision Tree` classifier has the least mean difference of -15% and an accuracy of 68.49%. The mean difference suggests that our model is bias towards Females, which means females are 15% more likely not to recidivate than males.

3. For Protected Attribute, Race, `Decision Tree` classifier has the least mean difference of 20.44% with an accuracy of 68.5%. Mean difference here suggests that, model is bias against African American, which means as per the model people of AA race 20% more likely to recidivate than people from other race.

In [42]:

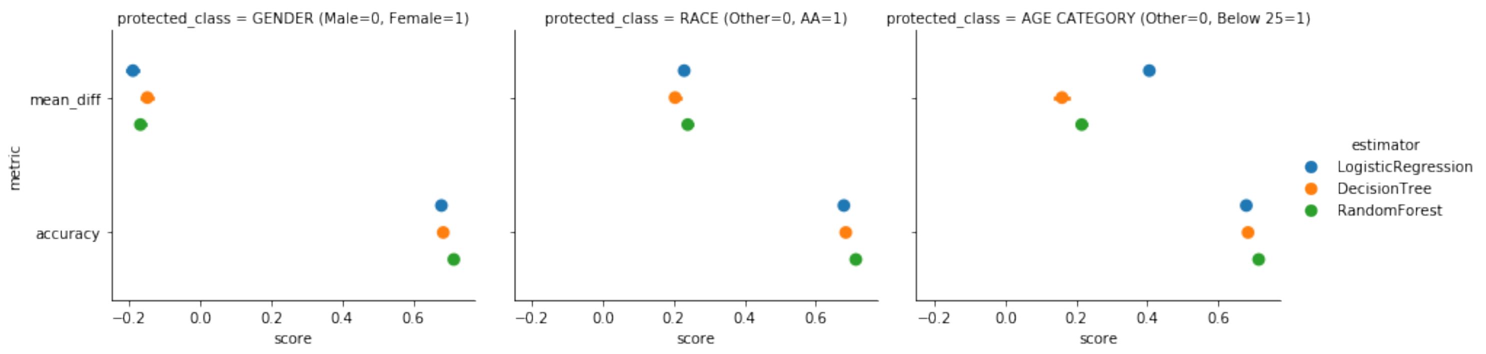
```
from themis_ml.metrics import mean_difference, normalized_mean_difference, \
    mean_confidence_interval
```

In [43]:

```
def plot_experiment_results(experiment_results):
    return (
        experiment_results
        .query("fold_type == 'test'")
        .drop(["fold_type", "cv_fold"], axis=1)
        .pipe(pd.melt, id_vars=["protected_class", "estimator"],
              var_name="metric", value_name="score")
        .pipe((sns.factorplot, "data"), y="metric",
              x="score", hue="estimator", col="protected_class", col_wrap=3,
              size=3.5, aspect=1.2, join=False, dodge=0.4))
```

In [44]:

```
plot_experiment_results(experiment_baseline);
```



Columns under consideration for further analysis

In [46]:

```
columns = ['juv_fel_count', 'decile_score', 'juv_misd_count', 'juv_other_count
',
                    'priors_count', 'days_b_screening_arrest', 'c_days_from_co
mpas', 'age',
                    'c_num_hours', 'num_hours_custody', 'crime_factor', 'score
_factor',
                    'is_Female', 'is_African-American', 'is_Less than 25']
```

Columns with no protected class

In [47]:

```
features_with_no_protected_class = ['juv_fel_count', 'decile_score', 'juv_misd
_count', 'juv_other_count',
                                    'priors_count', 'days_b_screening_arrest',
'c_days_from_compas', 'age',
                                    'c_num_hours', 'num_hours_custody', 'crime
_factor', 'score_factor']
```

Remove Protected Attribute (RPA)

Remove all protected attributes, Age Category, Gender and Race. This is naive approach in which we assume removal protected attribute removes the bias.

In [48]:

```
X = scaled_features[features_with_no_protected_class].values
```

In [49]:

```
y = scaled_features['will_not_recidivate'].values
```

In [50]:

```
s_female = scaled_features['is_Female'].values
s_black = scaled_features["is_African-American"].values
s_age_less_than_25 = scaled_features["is_Less than 25"].values
```

Using Logistic Regression, Decision Tree Classifier and Random Forest Classifer

In [51]:

```
LOGISTIC_REGRESSION = LogisticRegression(
    penalty="l2", C=0.001, class_weight="balanced")
DECISION_TREE_CLF = DecisionTreeClassifier(
    criterion="entropy", max_depth=10, min_samples_leaf=10, max_features=12,
    class_weight="balanced")
RANDOM_FOREST_CLF = RandomForestClassifier(
    criterion="entropy", n_estimators=50, max_depth=10, max_features=12,
    min_samples_leaf=10, class_weight="balanced")
estimators = [
    ("LogisticRegression", LOGISTIC_REGRESSION),
    ("DecisionTree", DECISION_TREE_CLF),
    ("RandomForest", RANDOM_FOREST_CLF)
]
```

In [52]:

```
experiment_rpa_female = cross_validation_experiment(
    estimators, X, y, s_female, "GENDER (Male=0, Female=1)")
experiment_rpa_race = cross_validation_experiment(
    estimators, X, y, s_black, "RACE (Other=0, AA=1)")
experiment_rpa_age = cross_validation_experiment(
    estimators, X, y, s_age_less_than_25, "AGE CATEGORY (Other=0, Below 25=1)"
)
```

```
Training models: protected_class = GENDER (Male=0, Female=1)
-------------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = RACE (Other=0, AA=1)
--------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = AGE CATEGORY (Other=0, Below 25
=1)
-------------------------------------------------------------------
---
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
```

In [53]:

```
experiment_rpa = pd.concat([
    experiment_rpa_female,
    experiment_rpa_race,
    experiment_rpa_age
])
experiment_rpa_summary = summarize_experiment_results(
    experiment_rpa)
experiment_rpa_summary.query("fold_type == 'test'")
```

Out[53]:

			mean_diff	accuracy
protected_class	**estimator**	**fold_type**		
AGE CATEGORY (Other=0, Below 25=1)	**DecisionTree**	**test**	0.148239	0.685001
	LogisticRegression	**test**	0.397939	0.675824
	RandomForest	**test**	0.210196	0.714383
GENDER (Male=0, Female=1)	**DecisionTree**	**test**	-0.122393	0.685422
	LogisticRegression	**test**	-0.092435	0.675824
	RandomForest	**test**	-0.133103	0.716003
RACE (Other=0, AA=1)	**DecisionTree**	**test**	0.194547	0.684871
	LogisticRegression	**test**	0.257471	0.675824
	RandomForest	**test**	0.227410	0.714415

Summary (RPA):

Removing all the protected attributes (Gender, Race, Age Category) we observe that,

1. For Protected Attribute, Age Category, `Decision Tree` classifier has the least point mean difference of 14.82% which is almost 1% less than Baseline (15.96%) and with same accuracy of 68.5%. But this result is still bias against people with age below 25, i.e., as per model, people with age less than 25 are 14.82% more likely to recidivate than people from other age group.

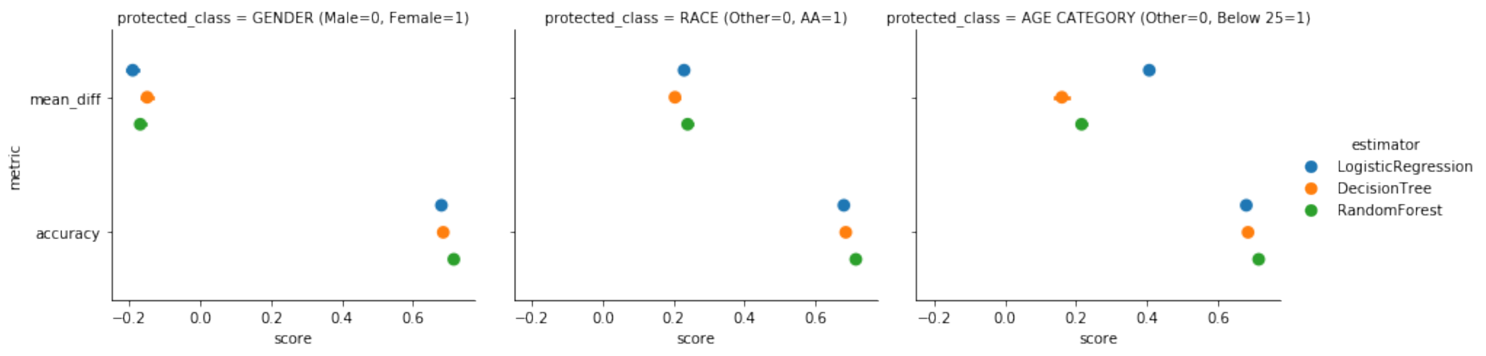
2. For Protected Attribute, Gender, `Logistic Regression` classifier has the least point mean difference of -9% which is quite improvement over -19% from Baseline with almost same accuracy of 67%. However, the mean difference suggests that our model is still bias towards Females, which means as per the model females 9% more likely not to recidivate than men.

3. For Protected Attribute, Race, `Decision Tree` classifier has the least point mean difference of 19.4% which is 1% improvement over 20.4% from Baseline with almost same accuracy of 68.5%. However, the mean difference suggests that our model is still bias against African American, which means people from AA race are 19.4% more likely to recidivate than people from other race.

Despite removing the protected attributes, our RPA approach didn't make a significant progess in mitigating bias. This means there is presence of either indirect prejudice or negative data.

```
In [54]:
```

```
plot_experiment_results(experiment_rpa);
```



Remove one protected attribute at a time

We are performing one more experiment with RPA. Instead of removing all Protected Attribute, here, we removed one and let the other persists and examine the mean difference and accuracy of the **removed** protected atrribute.

In [55]:

```
LOGISTIC_REGRESSION = LogisticRegression(
    penalty="l2", C=0.001, class_weight="balanced")
DECISION_TREE_CLF = DecisionTreeClassifier(
    criterion="entropy", max_depth=10, min_samples_leaf=10, max_features=14,
    class_weight="balanced")
RANDOM_FOREST_CLF = RandomForestClassifier(
    criterion="entropy", n_estimators=50, max_depth=10, max_features=14,
    min_samples_leaf=10, class_weight="balanced")
estimators = [
    ("LogisticRegression", LOGISTIC_REGRESSION),
    ("DecisionTree", DECISION_TREE_CLF),
    ("RandomForest", RANDOM_FOREST_CLF)
]
```

In [56]:

```
feature_no_gender = ['is_African-American', 'is_Less than 25','juv_fel_count',
'decile_score', 'juv_misd_count', 'juv_other_count',
                    'priors_count', 'days_b_screening_arrest', 'c_days_from_co
mpas', 'age',
                    'c_num_hours', 'num_hours_custody', 'crime_factor', 'score
_factor']
```

In [57]:

```
feature_no_race = ['is_Female', 'is_Less than 25','juv_fel_count', 'decile_sco
re', 'juv_misd_count', 'juv_other_count',
                    'priors_count', 'days_b_screening_arrest', 'c_days_from_co
mpas', 'age',
                    'c_num_hours', 'num_hours_custody', 'crime_factor', 'score
_factor' ]
```

In [58]:

```
feature_no_age = ['is_Female', 'is_African-American', 'juv_fel_count', 'decile
_score', 'juv_misd_count', 'juv_other_count',
                    'priors_count', 'days_b_screening_arrest', 'c_days_from_co
mpas', 'age',
                    'c_num_hours', 'num_hours_custody', 'crime_factor', 'score
_factor']
```

In [59]:

```
X_no_gender = scaled_features[feature_no_gender].values
X_no_race = scaled_features[feature_no_race].values
X_no_age = scaled_features[feature_no_age].values
```

In [60]:

```
experiment_naive_female = cross_validation_experiment(
    estimators, X_no_gender, y, s_female, "GENDER (Male=0, Female=1)")
experiment_naive_race = cross_validation_experiment(
    estimators, X_no_race, y, s_black, "RACE (Other=0, AA=1)")
experiment_naive_age_below_25 = cross_validation_experiment(
    estimators, X_no_age, y, s_age_less_than_25, "AGE CATEGORY (Other=0, Below
25=1)")
```

```
Training models: protected_class = GENDER (Male=0, Female=1)
------------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = RACE (Other=0, AA=1)
-------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = AGE CATEGORY (Other=0, Below 25
=1)
------------------------------------------------------------------
---
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
```

In [61]:

```
experiment_naive = pd.concat([
    experiment_naive_female,
    experiment_naive_race,
    experiment_naive_age_below_25
])
experiment_naive_summary = summarize_experiment_results(experiment_naive)
experiment_naive_summary.query("fold_type == 'test'")
```

Out[61]:

			mean_diff	accuracy
protected_class	**estimator**	**fold_type**		
AGE CATEGORY (Other=0, Below 25=1)	**DecisionTree**	**test**	0.159675	0.685423
	LogisticRegression	**test**	0.382130	0.681012
	RandomForest	**test**	0.212612	0.713831
GENDER (Male=0, Female=1)	**DecisionTree**	**test**	-0.120588	0.685488
	LogisticRegression	**test**	-0.092648	0.675078
	RandomForest	**test**	-0.136762	0.713409
RACE (Other=0, AA=1)	**DecisionTree**	**test**	0.193286	0.685001
	LogisticRegression	**test**	0.259633	0.679002
	RandomForest	**test**	0.228427	0.715873

Summary (naive RPA):

Removing one protected attribute (Gender, Race, Age Category) at a time, we observe that,

1. For Protected Attribute, Age Category, `Decision Tree` classifier has the least point mean difference of 16% which is same as 16% from Baseline with same accuracy of 68.5%. No improvement. However, the mean difference suggests our model is still bias against age category Less than 25, which means people below 25 are 16% more likely to recidivate.

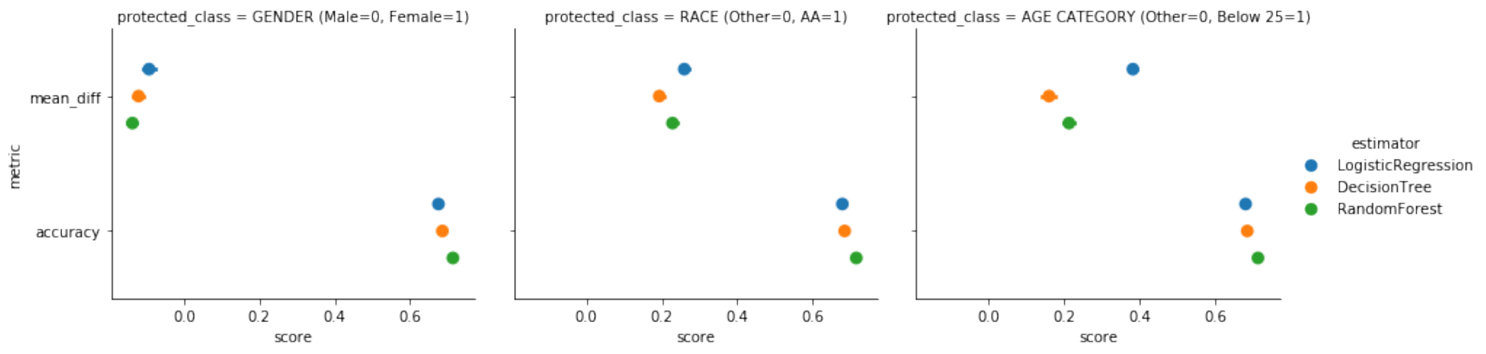
2. For Protected Attribute, Gender, `Logistic Regression` classifier has the least point mean difference of -9% which is quite improvement from Baseline (-19%) with almost same accuracy of 67% with slight reduction around 0.5%. However, the mean difference suggests that our model is still bias towards Females, which means as per the model females are 9% more likely not to recidivate.

3. For Protected Atrribute, Race, `Decision Tree` classifier has the least point mean difference of 19.4% which is almost 1% less than Baseline but with same accuracy of 68%. But this result is still bias against African American, i.e., as per model, African American are 19% more likely to recidivate.

This results are same as RPA with protected attributes removed.
Despite removing the protected attributes, our naive approach didn't make a significant progess in mitigating bias. This means there is presence of either indirect prejudice or underestimation.

In [62]:

```
plot_experiment_results(experiment_naive);
```



Relabelling/ Relabel Target Variable (RTV)

Relabelling **[2]**, also called *Massaging*, modifies y_{train} by relabelling the target variables in such a way that it **promotes** high rank members of the disadvantaged protected class (e.g. “females” in case of Gender) and **demotes** lower rank members of the advantaged class (e.g.“males” in case of Gender).

A ranker R (e.g. logistic regression) is trained on Dataset D, and ranks are generated for all observations. Some of the top-ranked observations X_{d,y^-} are “promoted” to X_{d,y^+} and some of the bottom-ranked observations X_{a,y^+} are “demoted” to X_{a,y^-} such that the proportion of y^+ are equal in both X_d and X_a. Two caveats of this method are that it is intrusive because it directly manipulates y, and that it narrowly defines fairness as the uniform distribution of benefits between X_a and X_d.

In [63]:

```
LOGISTIC_REGRESSION = LogisticRegression(
    penalty="l2", C=0.001, class_weight="balanced")
DECISION_TREE_CLF = DecisionTreeClassifier(
    criterion="entropy", max_depth=10, min_samples_leaf=10, max_features=15,
    class_weight="balanced")
RANDOM_FOREST_CLF = RandomForestClassifier(
    criterion="entropy", n_estimators=50, max_depth=10, max_features=15,
    min_samples_leaf=10, class_weight="balanced")
estimators = [
    ("LogisticRegression", LOGISTIC_REGRESSION),
    ("DecisionTree", DECISION_TREE_CLF),
    ("RandomForest", RANDOM_FOREST_CLF)
]
```

In [64]:

```
from sklearn.base import clone

from themis_ml.preprocessing.relabelling import Relabeller
from themis_ml.meta_estimators import FairnessAwareMetaEstimator

relabeller = Relabeller()
relabelling_estimators = [
    (name, FairnessAwareMetaEstimator(e, relabeller=relabeller))
    for name, e in estimators]

experiment_relabel_female = cross_validation_experiment(
    relabelling_estimators, b_X, y, s_female, "GENDER (Male=0, Female=1)")
experiment_relabel_race = cross_validation_experiment(
    relabelling_estimators, b_X, y, s_black, "RACE (Other=0, AA=1)")
experiment_relabel_age_below_25 = cross_validation_experiment(
    relabelling_estimators, b_X, y, s_age_less_than_25, "AGE CATEGORY (Other=0
, Below 25=1)")
```

```
Training models: protected_class = GENDER (Male=0, Female=1)
-------------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = RACE (Other=0, AA=1)
--------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = AGE CATEGORY (Other=0, Below 25
=1)
------------------------------------------------------------------
---
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
```

In [65]:

```
experiment_relabel = pd.concat([
    experiment_relabel_female,
    experiment_relabel_race,
    experiment_relabel_age_below_25
])
experiment_relabel_summary = summarize_experiment_results(experiment_relabel)
experiment_relabel_summary.query("fold_type == 'test'")
```

Out[65]:

			mean_diff	**accuracy**
protected_class	**estimator**	**fold_type**		
AGE CATEGORY (Other=0, Below 25=1)	**DecisionTree**	**test**	0.083084	0.689509
	LogisticRegression	**test**	0.065746	0.686848
	RandomForest	**test**	0.083962	0.712567
GENDER (Male=0, Female=1)	**DecisionTree**	**test**	-0.910055	0.560106
	LogisticRegression	**test**	-0.667045	0.619522
	RandomForest	**test**	-0.915254	0.558517
RACE (Other=0, AA=1)	**DecisionTree**	**test**	0.107460	0.679455
	LogisticRegression	**test**	0.118071	0.673619
	RandomForest	**test**	0.146008	0.702611

Summary Relabel

Following observations are made after applying relabelling fairness-aware method -

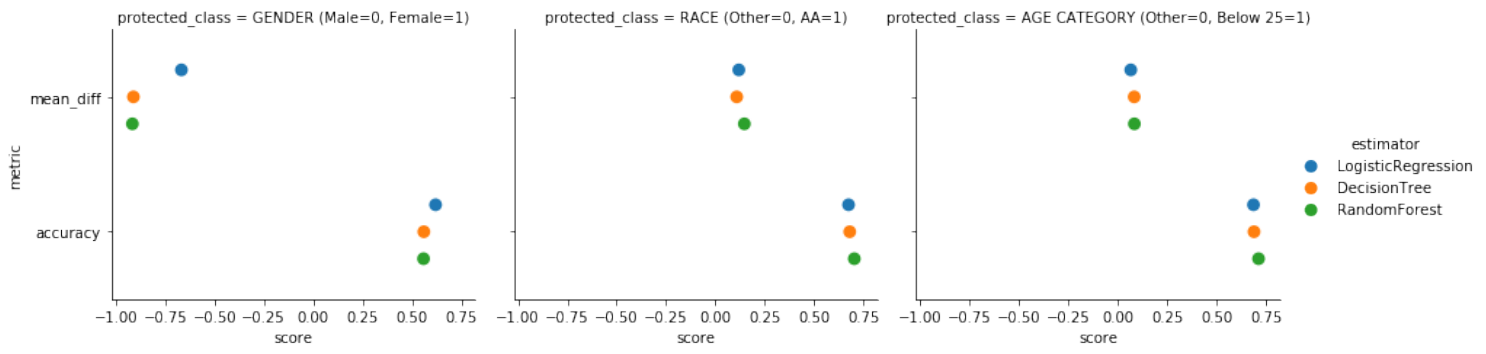
1. For Protected Attribute, Age Category, `Logistic Regression` classifier has the least point mean difference of 6.5% which is significantly lower than Baseline (40.6%) of `Logistic Regression` and Baseline best `Decision Tree` (16%) with almost same accuracy of 68%. Model performed better in reducing the bias now we can state that, people below 25 are 6.5% more likely to recidivate than other age group

2. For Protected Attribute, Gender, `Logistic Regression` classifier has the least point mean difference of -6.67% which is quite improvement from Baseline(-19%) with a reduction in accuracy to 61.9% from 67.9%. We achieved reduction in mean difference but at the expense of accuracy. So, we observe there is a trade off. However, the mean difference suggests that our model is still bias towards Females, which means females are 6.67% more likely not to recidivate than men.

3. For Protected Atrribute, Race, `Decision Tree` classifier has the least point mean difference of 10.7% which is 9.5% less than Baseline with almost same accuracy of 68%. There is an improvement in reducing the bias, still model is bias against African American, i.e., African American are 10.7% more likely to recidivate than people from other race.

In [66]:

```
plot_experiment_results(experiment_relabel);
```



Additive Counterfactually Fair Model (ACF)

Additive Counterfactually Fair (ACF) **[4]** model is a method described within the framework of counterfactual fairness. The main idea is to train linear models to predict each feature using the protected class attribute(s) as input. We can then compute the residuals ε_{ij} between the predicted feature values and true feature values for each observation *i* and each feature *j*. The final model is then trained on ε_{ij} as features to predict *y*.

In [67]:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor

from themis_ml.linear_model.counterfactually_fair_models import LinearACFClass
ifier

linear_acf_estimators = [
    (name, LinearACFClassifier(
         target_estimator=e,
         binary_residual_type="absolute"))
    for name, e in estimators]

experiment_acf_female = cross_validation_experiment(
    linear_acf_estimators, b_X, y, s_female, "GENDER (Male=0, Female=1)")
experiment_acf_race = cross_validation_experiment(
    linear_acf_estimators, b_X, y, s_black, "RACE (Other=0, AA=1)")
experiment_acf_age_below_25 = cross_validation_experiment(
    linear_acf_estimators, b_X, y, s_age_less_than_25, "AGE CATEGORY (Other=0,
Below 25=1)")
```

```
Training models: protected_class = GENDER (Male=0, Female=1)
--------------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = RACE (Other=0, AA=1)
---------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = AGE CATEGORY (Other=0, Below 25
=1)
-------------------------------------------------------------------
---
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
```

In [68]:

```
experiment_acf = pd.concat([
    experiment_acf_female,
    experiment_acf_race,
    experiment_acf_age_below_25
])
experiment_acf_summary = summarize_experiment_results(experiment_acf)
experiment_acf_summary.query("fold_type == 'test'")
```

Out[68]:

			mean_diff	**accuracy**
protected_class	**estimator**	**fold_type**		
AGE CATEGORY (Other=0, Below 25=1)	**DecisionTree**	**test**	0.197065	0.679975
	LogisticRegression	**test**	0.063659	0.683379
	RandomForest	**test**	0.237198	0.713020
GENDER (Male=0, Female=1)	**DecisionTree**	**test**	-0.162278	0.674657
	LogisticRegression	**test**	0.018620	0.675532
	RandomForest	**test**	-0.180487	0.710587
RACE (Other=0, AA=1)	**DecisionTree**	**test**	0.191452	0.673780
	LogisticRegression	**test**	0.023626	0.670599
	RandomForest	**test**	0.242515	0.704394

Summary Additive Counterfactually Fair Model

Following observations are made after applying ACF -

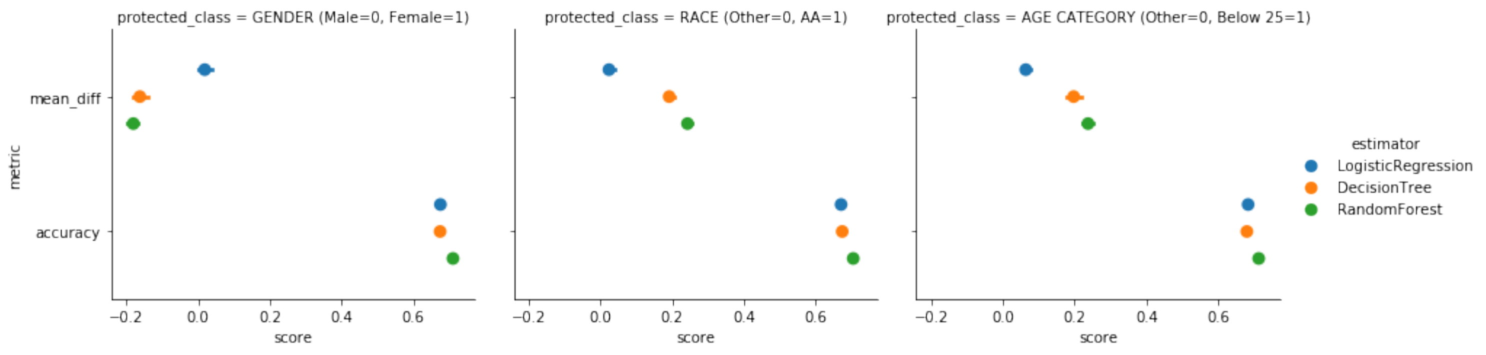
1. For Protected Attribute, Age Category, `Logistic Regression` classifier has the least point mean difference of 6.3% which is almost 10% lower than Baseline best `Decision Tree`, 16% and significantly less for `Logistic Regression` of 40%. Accuracy is not compromised, its 68%. Model performed better in reducing the bias, now we can state that, people below 25 are 6.3% more likely to recidivate than other age group

2. For Protected Attribute, Gender, `Logistic Regression` classifier has the least point mean difference of 1.8% which is significant improvement over -19% from Baseline for `Logistic Regression` and Baseline best `Decision Tree` (15%). Also, accuracy is almost same around 67%. Here we achieved good reduction in mean difference but now mean difference is positive, which means, model is bias against Females, which means females are 1.8% more likely to recidivate than men.

3. For Protected Atrribute, Race, `Logistic Regression` classifier has the least point mean difference of 2.3% which is almost 20% less than Baseline `Logistic Regression` (22.9%) and Baseline best `Decision Tree` (20%). However, observed slight reduction in accuracy from 68% in Baseline to 67% in ACF. From this results we can say that, African American are 2.3% more likely to recidivate than people from other race.

In [69]:

```
plot_experiment_results(experiment_acf);
```



Reject-option Classification (ROC)

Unlike Relabelling and ACF methods outlined above, ROC **[5]** do not modify the training data or the training process. Rather, they postprocess predictions in a way that reduces potentially discriminatory (PD) predictions.

ROC works by training an initial classifier on a dataset, say , D, generating predicted probabilities on the test set, and then computing the proximity of each prediction to the decision boundary learned by the classifier. Within this boundary defined by the critical region threshold θ, where $0.5 < \theta < 1$, X_d are assigned as y^+ and X_a are assigned as y^-.

In [70]:

```
from themis_ml.postprocessing.reject_option_classification import \
    SingleROClassifier

single_roc_clf_estimators = [
    (name, SingleROClassifier(estimator=e))
    for name, e in estimators]

experiment_single_roc_female = cross_validation_experiment(
    single_roc_clf_estimators, b_X, y, s_female, "GENDER (Male=0, Female=1)")
experiment_single_roc_race = cross_validation_experiment(
    single_roc_clf_estimators, b_X, y, s_black, "RACE (Other=0, AA=1)")
experiment_single_roc_age_below_25 = cross_validation_experiment(
    single_roc_clf_estimators, b_X, y, s_age_less_than_25, "AGE CATEGORY (Othe
r=0, Below 25=1)")
```

```
Training models: protected_class = GENDER (Male=0, Female=1)
------------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = RACE (Other=0, AA=1)
-------------------------------------------------------
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
Training models: protected_class = AGE CATEGORY (Other=0, Below 25
=1)
------------------------------------------------------------------
---
LogisticRegression, fold:
DecisionTree, fold:
RandomForest, fold:
```

In [71]:

```
experiment_single_roc = pd.concat([
    experiment_single_roc_female,
    experiment_single_roc_race,
    experiment_single_roc_age_below_25
])
experiment_single_roc_summary = summarize_experiment_results(
    experiment_single_roc)
experiment_single_roc_summary.query("fold_type == 'test'")
```

Out[71]:

			mean_diff	**accuracy**
protected_class	**estimator**	**fold_type**		
AGE CATEGORY (Other=0, Below 25=1)	**DecisionTree**	**test**	0.164327	0.676178
	LogisticRegression	**test**	-0.164299	0.562315
	RandomForest	**test**	0.207492	0.694630
GENDER (Male=0, Female=1)	**DecisionTree**	**test**	-0.134830	0.675887
	LogisticRegression	**test**	-0.097920	0.562315
	RandomForest	**test**	-0.157383	0.693887
RACE (Other=0, AA=1)	**DecisionTree**	**test**	0.195441	0.675984
	LogisticRegression	**test**	0.116451	0.562315
	RandomForest	**test**	0.222429	0.694017

Summary Reject Option Classification Model

Following observations are made after applying ACF -

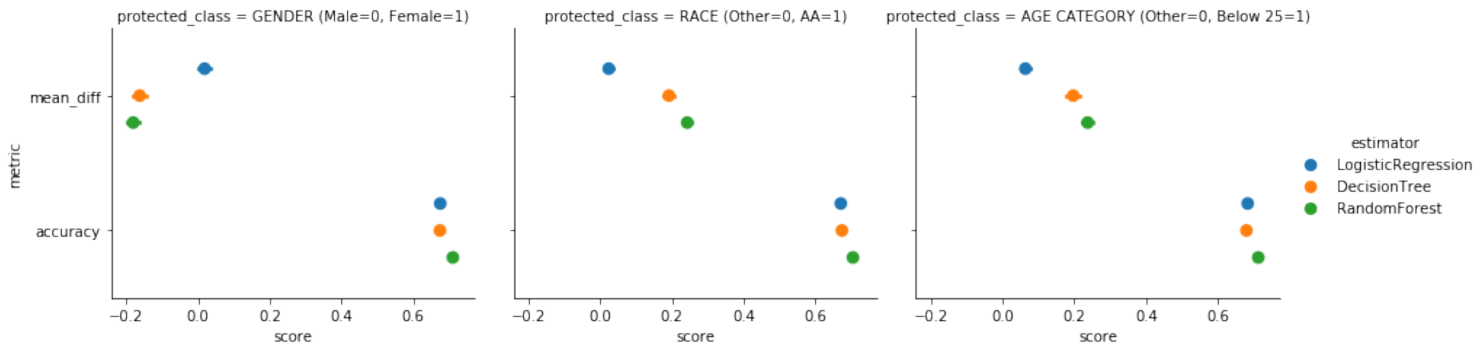
1. For Protected Attribute, Age Category, `Decision Tree` classifier has the least point mean difference of 16.3% which is almost same as than Baseline best `Decision Tree`, 16% (0.4% higher). Accuracy is reduced by 1%; Baseline accuracy = 68.5% and ROC accuracy = 67.5%. No significant improvement, rather slight reduction in mean difference and accuracy. This result shows, people with age less than 25 are are 16.3% more like to recidivate than people with age >=25.

2. For Protected Attribute, Gender, `Logistic Regression` classifier has the least point mean difference of -9% which is significant improvement over -19% from Baseline. But, accuracy is significantly compromised, it is reduced from 67.98% to 56.23%. This results shows a not so good fairness utility trade-off. However, the mean difference suggests that our model is still bias towards Females, which means females are 9% more likely not to recidivate than men.

3. For Protected Atrribute, Race, `Logistic Regression` classifier has the least point mean difference of 11.6% which is almost 10.5% less than Baseline `Logistic Regression` (22.9%) and Baseline best `Decision Tree` (20%). However, we observed reduction in accuracy from 68% in Baseline to 56% in ROC. From this results we can say that, African American are 11.6% more likely to recidivate than people from other race.

In [72]:

```
plot_experiment_results(experiment_acf);
```



Compare all results

In [73]:

```
compare_experiments = (
    pd.concat([
        experiment_baseline.assign(experiment="B"),
        experiment_naive.assign(experiment="RPA"),
        experiment_relabel.assign(experiment="RTV"),
        experiment_acf.assign(experiment="ACF"),
        experiment_single_roc.assign(experiment="ROC")
    ])
    .assign(
        protected_class=lambda df: df.protected_class.str.replace("_", " "),
    )
)
METRICS_COLUMNS = [
    "protected_class", "estimator", "mean_diff", "accuracy"]
```

In [74]:

```
compare_experiments.head(2)
```

Out[74]:

	protected_class	estimator	cv_fold	fold_type	mean_diff	accuracy	exp
0	GENDER (Male=0, Female=1)	LogisticRegression	0	train	-0.196466	0.678919	B
1	GENDER (Male=0, Female=1)	LogisticRegression	0	test	-0.166698	0.700162	B

In [75]:

```
def summarize_overall_results(experiment_df):
    return (
        experiment_df
        .drop("cv_fold", axis=1)
        .groupby(["protected_class", "estimator", "fold_type","experiment"])
        .mean())

overall_results = summarize_overall_results(compare_experiments)
result = overall_results.query("fold_type == 'test'")
result
```

Out[75]:

				mean_diff	accuracy
protected_class	**estimator**	**fold_type**	**experiment**		
AGE CATEGORY (Other=0, Below 25=1)	**DecisionTree**	**test**	**ACF**	0.197065	0.679975
			B	0.159953	0.684645

			ROC	0.164327	0.676178
			RPA	0.159675	0.685423
			RTV	0.083084	0.689509
	LogisticRegression	**test**	**ACF**	0.063659	0.683379
			B	0.406302	0.679812
			ROC	-0.164299	0.562315
			RPA	0.382130	0.681012
			RTV	0.065746	0.686848
	RandomForest	**test**	**ACF**	0.237198	0.713020
			B	0.215616	0.714805
			ROC	0.207492	0.694630
			RPA	0.212612	0.713831
			RTV	0.083962	0.712567
GENDER (Male=0, Female=1)	**DecisionTree**	**test**	**ACF**	-0.162278	0.674657
			B	-0.149723	0.684937
			ROC	-0.134830	0.675887
			RPA	-0.120588	0.685488
			RTV	-0.910055	0.560106
	LogisticRegression	**test**	**ACF**	0.018620	0.675532
			B	-0.190379	0.679812
			ROC	-0.097920	0.562315
			RPA	-0.092648	0.675078
			RTV	-0.667045	0.619522
	RandomForest	**test**	**ACF**	-0.180487	0.710587
			B	-0.169214	0.714706
			ROC	-0.157383	0.693887
			RPA	-0.136762	0.713409
			RTV	-0.915254	0.558517
RACE (Other=0, AA=1)	**DecisionTree**	**test**	**ACF**	0.191452	0.673780
			B	0.203724	0.685261
			ROC	0.195441	0.675984
			RPA	0.193286	0.685001
			RTV	0.107460	0.679455
	LogisticRegression	**test**	**ACF**	0.023626	0.670599

			B	0.229384	0.679812
			ROC	0.116451	0.562315
			RPA	0.259633	0.679002
			RTV	0.118071	0.673619
	RandomForest	**test**	**ACF**	0.242515	0.704394
			B	0.239449	0.713637
			ROC	0.222429	0.694017
			RPA	0.228427	0.715873
			RTV	0.146008	0.702611

Overall Best Model Observed

```
In [77]:
```

```
overall_best_model = result.groupby('protected_class').apply(lambda x: x[x.mea
n_diff ==abs(x.mean_diff).min()])
overall_best_model
```

```
Out[77]:
```

					mean
protected_class	**protected_class**	**estimator**	**fold_type**	**experiment**	
AGE CATEGORY (Other=0, Below 25=1)	**AGE CATEGORY (Other=0, Below 25=1)**	**LogisticRegression**	**test**	**ACF**	0.063
GENDER (Male=0, Female=1)	**GENDER (Male=0, Female=1)**	**LogisticRegression**	**test**	**ACF**	0.018
RACE (Other=0, AA=1)	**RACE (Other=0, AA=1)**	**LogisticRegression**	**test**	**ACF**	0.023

Summarized results

`summarized_overall_results` shows mean value of *mean_difference and accuracy* for each type of fairness aware method per estimator for each protected attribute/ class.

Naive approach of removing the protected attribute won't contribute much towards reducing the mean difference. This indicates either presence of indirect prejudice or underestimation or negative legacy.

All the three protected class (Age Category, Gender, Race) have lowest bias factor with *Logistic Regression* classifier applied with Additive Counterfactually Fairness Model(ACF).

ACF has least mean difference of [0.064, 0.019, 0.023] for protected class [Age Category, Gender, Race] respectively, which is significantly lower than baseline best (decision trees) [0.16, -0.15, 0.204] by [60%, 87%, 88%] respectively.

We also observed that with ACF we achieved almost same accuracy as Baseline ~68%

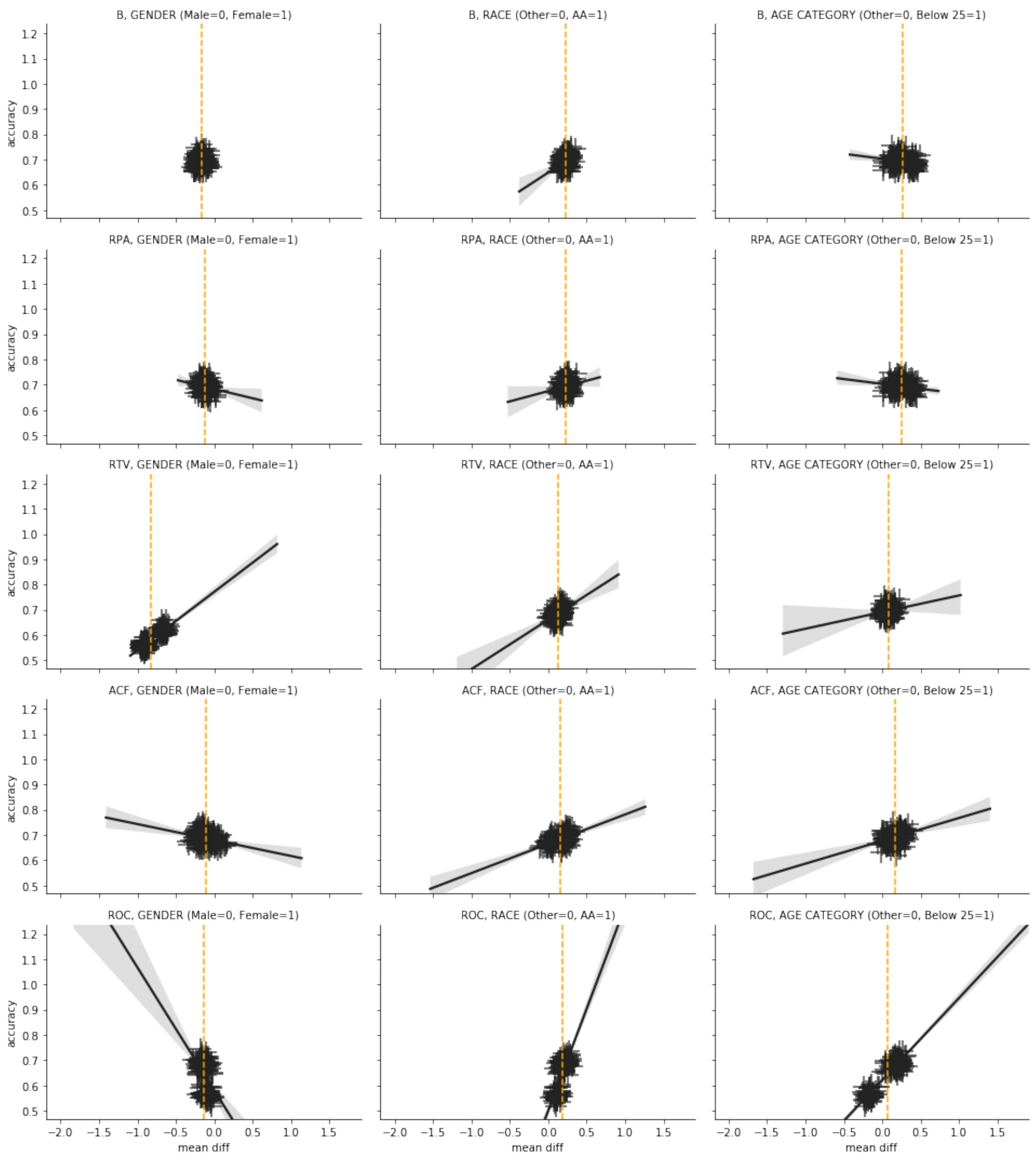
Fairness Utility Trade-Off

In [79]:

```
my_palette = sns.color_palette(["#222222"])

def plot_utility_fairness_tradeoff(x, y, **kwargs):
    ax = plt.gca()
    data = kwargs.pop("data")
    mean_x = data[x].mean()
    # mean_y = data[y].mean()
    sns_ax = sns.regplot(x=x, y=y, data=data, marker = '+', scatter_kws={'s':4
00},
                       **kwargs)
    sns_ax.axvline(mean_x, color='orange', linestyle='--')
    # sns_ax.axhline(mean_y, color='orange', linestyle='--')
    bottom_padding = 0.05
    top_padding = 0.5
    ylim = (data[y].min() - bottom_padding, data[y].max() + top_padding)
    sns_ax.set_ylim(*ylim)

g = sns.FacetGrid(
    (
        compare_experiments
        .drop("cv_fold", axis=1)
        .reset_index()
        .query("fold_type == 'test'")
        .rename(
            columns={"mean_diff": "mean diff"})
    ),
    col="protected_class",
    row="experiment",
    hue="experiment",
    size=3.0, aspect=1.5, sharey=True,
    palette=my_palette)
g.map_dataframe(plot_utility_fairness_tradeoff, "mean diff", "accuracy")
g.set_titles(template="{row_name}, {col_name}")
g.fig.tight_layout()
```



From the above results we observe that,

There exists a relationship between mean difference and accuracy. So, if we try to want to achieve a completely fair model, with mean difference = 0.0 then it will come with the expense of accuracy.

References

[1] https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm (https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm).

[2] Niels Bantilan, "*Themis-ml: A Fairness-aware Machine Learning Interface for End-to-end Discrimination Discovery and Mitigation,*" arXiv 1710.06921v1 [cs.CY] 18 Oct 2017

[3] I. Zliobaite, *"A survey on measuring indirect discrimination in machine learning,"* arXiv preprint arXiv:1511.00148, 2015.

[4] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva, *"Counterfactual fairness,"* arXiv preprint arXiv:1703.06856, 2017

[5] F. Kamiran, A. Karim, and X. Zhang, *"Decision theory for discrimination-aware classification,"* in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pp. 924–929, IEEE, 2012.